TUGAS AKHIR – TE 145561

DESAIN PROTOTIPE DAN KOORDINASI DIGITAL RELAY SATU FASA TERHADAP GANGGUAN OVERCURRENT

Dibyo Sindhu Pradana
NRP 2214038018

Dosen Pembimbing
Dr. Eng. Ardyono Priyadi, S.T., M.Eng.

PROGRAM STUDI TEKNIK LISTRIK
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

-----Halaman ini sengaja dikosongkan-----

FINAL PROJECT – TE 145561

PROTOTYPE DESIGN AND COORDINATION OF DIGITAL RELAY ONE PHASE ON OVERCURRENT DISORDERS

Dibyo Sindhu Pradana
NRP 2214038018

Advisor
Dr. Eng. Ardyono Priyadi, S.T., M.Eng.

ELECTRICAL ENGINEERING STUDY PROGRAM
Electrical and Automation Engineering Department
Vocational Faculty
Institut Teknologi Sepuluh Nopember
Surabaya 2017

-----Halaman ini sengaja dikosongkan-----

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "DESAIN PROTOTIPE DAN KOORDINASI DIGITAL RELAY SATU FASA TERHADAP GANGGUAN OVERCURRENT" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, **19 Juli 2017**

Mahasiswa

Dibyo Sindhu Pradana
NRP 2214038018

-----Halaman ini sengaja dikosongkan-----

DESAIN PROTOTIPE DAN KOORDINASI DIGITAL RELAY SATU FASA TERHADAP GANGGUAN OVERCURRENT

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Ahli Madya
Pada
Program Studi Teknik Listrik
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing

Dr. Eng. Ardyono Priyadi, S.T., M.Eng.
NIP. 1973 09 27 1998 03 1004

SURABAYA
JULI, 2017

-----Halaman ini sengaja dikosongkan-----

DESAIN PROTOTIPE DAN KOORDINASI DIGITAL RELAY SATU FASA TERHADAP GANGGUAN OVERCURRENT

Nama : Dibyo Sindhu Pradana
Pembimbing : Dr. Eng. Ardyono Priyadi, S.T., M.Eng.

ABSTRAK

Dalam suatu sistem penyaluaran tenaga listrik yang memiliki skala besar seperti PT PLN Persero komponen komponen utama maupun bantu haruslah di perhitungkan dengan baik dan se efisien mungkin . salah satunya yang sangat di perhitungkan dalam proses penyaluran energi listrik adalah sistem proteksi. Keandalan dalam sistem koordinasi sangatlah di perlukan guna memperkecil pengaruh gangguan terlebih lagi dengan perkembangan jumlah pembangkit dan kapasitansi gardu induk , kegagalan dalam sistem proteksi sangatlah berdampak besar terhadap proses penyaluran energi listrik.

Beberapa faktor yang menjadi kendalan dalam sistem proteksi adalah salahnya pengkoordinasian relay dan keterbatasan jumlah sumber daya manusia ahli yang dapat memperhitungkan dan memproses koordinasi proteksi. Kendala jarak,waktu dan salahnya koordinasi pada relay dapat berakibat memperluasnya lokasi gangguan sehingga penggunaan energi listrik tidak efisien. Untuk itu di perlukan suatu sistem baru guna mengurangi kendala tersebut sehingga lebih efektif.

Salah satu cara yaitu dengan sistem pengaturan dan koordinasi relay jarak jauh yang dapat di lakukan pada lokasi yang sama. Sehingga kendala jarak ,waktu dan sumber daya ahli dapat di kurangi. Proses monitoring juga dapat di fungsikan sebagai pemberi tahu kondisi jaringan pada saat itu

Berdasarkan hal tersebut maka kami membuat Tugas akhir berupa relay digital yang dapat mengirim dan menerima data pengaturan dan kondisi jaringan. Fungsi dari relay tersebut sebagai simulasi koordinasi dan monitoring sistem proteksi.Proses pangaturan koordinasi menggunakan sebuah computer yang terhubung dengan kedua relay. Program yang di gunakan adalah Lazarus yang komunikasinya di hubungkan dengan kedua relai menggunakan protokol Modbus.

Kata Kunci : sistem proteksi, digital relay, monitoring sistem proteksi,

-----Halaman ini sengaja dikosongkan-----

PROTOTYPE DESIGN AND COORDINATION OF DIGITAL RELAY ONE PHASE ON OVERCURRENT DISORDERS

Name : Dibyo Sindhu Pradana
Advisor : Dr. Eng. Ardyono Priyadi, S.T., M.Eng.

ABSTRACT

In a large scale electric power distribution system such as PT PLN Persero, the main component components as well as aids should be calculated as well and efficiently as possible. One of which is very calculated in the process of electrical energy distribution is a protection system. Reliability in the coordination system is needed to minimize the effect of disturbance, especially with the development of the number of substations and the substation capacitance, failure in the protection system is very big impact on the process of electrical energy distribution.

Some of the factors that become a constraint in the protection system is the wrong coordination of the relay and the limited number of human resources experts who can calculate and process the coordination of protection. The distance, time constraints and incorrect coordination of the relay can result in expanding the disturbance location so that the use of electrical energy is inefficient. For that in need of a new system to reduce the constraints so more effective.

One way is with the system of remote relay coordination and coordination that can be done at the same location. So that distance constraints, time and expert resources can be reduced. Monitoring process can also be enabled as a giver of network conditions at that time.

Based on that we then make the final task of a digital relay that can send and receive data settings and network conditions. The function of the relay is as a simulation of coordination and monitoring of the protection system. The coordination coordination process uses a computer connected to the two relays. The program used is Lazarus whose communications are connected to both relays using the Modbus protocol.

Keywords :*Protection system, digital relay, monitoring protection system,*

-----Halaman ini sengaja dikosongkan-----

KATA PENGANTAR

Alhamdulillah kami panjatkan kepada Allah Subhanahu Wa Ta'ala, atas limpahan rahmat dan kemudahan dariNya, hingga kami dapat menyelesaikan tugas akhir ini dengan baik, begitu pula dengan pembuatan buku tugas akhir ini.

Tugas akhir ini dilakukan untuk memenuhi beban satuan kredit semester (SKS) yang harus ditempuh sebagai persyaratan akademis di Jurusan D3 Teknik Elektro Institut Teknologi Sepuluh Nopember Surabaya untuk menyelesaikan program pendidikan Diploma di Teknik Elektro dengan judul "**DESAIN PROTOTIPE DAN KOORDINASI DIGITAL RELAY SATU FASA TERHADAP GANGGUAN OVERCURRENT**" Penulis mengucapkan terima kasih kepada Ibu dan Bapak penulis yang memberikan berbagai bentuk doa serta dukungan tulus tiada henti, Bapak Dr. Eng. Ardyono Priyadi, S.T., M.Eng. atas segala bimbingan ilmu, moral, dan spiritual dari awal hingga terselesaikannya Tugas Akhir ini. Penulis juga mengucapkan banyak terima kasih kepada semua pihak yang telah membantu baik secara langsung maupun tidak langsung dalam proses penyelesaian Tugas Akhir ini.

Penulis menyadari dan memohon maaf atas segala kekurangan pada Tugas Akhir ini. Akhir kata, semoga Tugas Akhir ini dapat bermanfaat dalam pengembangan keilmuan di kemudian hari.

Surabaya, **19 Juli 2017**

Penulis

-----Halaman ini sengaja dikosongkan-----

DAFTAR ISI

-----Halaman ini sengaja dikosongkan-----

DAFTAR GAMBAR

HALAMAN

DAFTAR TABEL

HALAMAN

-----Halaman ini sengaja dikosongkan-----

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam sistem penyaluran energi listrik terbagi beberapa sistem sistem kecil yang saling mendukung agar proses penyaluran energy listrik dapat tersalurkan dengan baik dan handal. Untuk itu di perlukansustun sistem proteksi yang baik agar sistem tersebut dapat Relai arus lebih (*Over Current Relay*) mendapatkan peran yang sangat penting pada sistem proteksi saluran tenaga listrik. Sistem proteksi ini membutuhkan keandalan yang tinggi untuk menjaga keamanan pada suatu sistem. Koordinasi waktu antar relai pada saluran merupakan hal yang sangat penting pada sistem proteksi agar tidak terjadinya salah pengoperasian. Selain itu, relai pengaturan arus yang di tetapkan haruslah sesuai dengan koordinasi sehingga sistem proteksi dapat bekerja secara optimal.

Dalam suatu sistem yang selalu berkembang, perubahan-perubahan kondisi yang terjadi akan selalu ada. Begitu juga dengan sistem kelistrikan. Kebutuhan energi listrik pada tahun 2025 akan menjadi 457 TWh, atau tumbuh rata-rata sebesar 8,6% per tahun untuk periode tahun 2016- 2025. Sedangkan beban puncak *non coincident* pada tahun 2025 akan menjadi 74.383 MW atau tumbuh rata-rata 8,4% per tahun.[1]

Dengan pertumbuhan masyarakat saat ini yang sangat pesat, kebutuhan beban dari sistem kelistrikan pun akan terus meningkat. Namun pertambahan beban ini juga harus diiringi dengan pertambahan sumber. Agar beban yang semakin besar dapat tersuplai listrik. Pertambahan sumber ini menyebabkan masalah lain, yaitu pengkoordinasian setting pada alat proteksi. Jika sumber bertambah, maka setting yang ada juga harus berubah karena arus yang mengalir bertambah besar.

Perubahan besarnya arus ini disebabkan oleh bertambahnya beban yang harus disuplai oleh sumber. Sehingga jika terdapat perubahan pada sumber, otomatis *setting* relai juga berubah. Namun selama ini yang terjadi dilapangan adalah *setting* relai yang masih dilakukan secara manual, sehingga operator harus men-*setting* relai agar sesuai dengan besarnya sumber. Dan jika terdapat kesalahan *setting* saat sumber berubah, maka akan berpengaruh pada sistem pengamanan saluran.

Oleh karena itu, pada tugas akhir ini bertujuan untuk membuat sebuh permodelan yang nantinya dapat digunakan untuk mengetahui pengkoordinasikan relai yang memiliki dua sumber. *Setting* relai harus

dapat bersifat adaptif, yaitu dapat mengubah *setting* sesuai dengan kondisi sumber (saat sumber bertambah). Perubahan *setting* ini akan diatur oleh program yang telah dimasukkan ke *microcontroller*. Permodelan relai ini menggunakan *microcontroller* yang digunakan sebagai media *sensing* arus dan sebagai media *setting* dan koordinasi relai untuk nantinya jika ada gangguan akan mengirim sinyal *trip* pada kontaktor.

1.2 Permasalahan

Adapun permasalahan yang akan kami angkat sebagai bahan Tugas Akhir ini :

Perubahan kapasitas dari suatu sistem penyaluran seringkali di lakukan sehinga pengaturan relay sering berubah ubah, dalam proses perubahan pengaturan relay, para teknisi di haruskan melakukan pengaturan secara manual di mana relay terpasang. Maka dari itu, pada tugas akhir ini akan membuat permodelan *setting Over Current Relay Secara Scada* dengan menggunakan Arduino MEGA sebagai pemrosesan dan *RS485* sebagai media komunikasi sehingga pengaturan dan pemantauan relay dapat di lakukan dari jarak jauh pada satu lokasi.

1.3 Batasan Masalah

Agar dalam penulisan buku Tugas Akhir ini tidak menyimpang dan melebar jauh dari tujuan yang semula direncanakan dibuat di buat beberapa acuan khusus dalam pembahasan , Adapun batasan batasan yang menjadi lingkup masalah yang ditetapkan dalam buku tugas akhir ini sebagai berikut :

a. Pengaturan nilai arus dan waktu yang di tetapkan pada tiap rele
b. Karakteristik jenis proteksi gangguan arus hubung singkat dan beban lebih berdasarkan waktu kerja rele
c. Simulasi proteksi jaringan rendah satu fasa

1.4 Tujuan

Pembuatan Permodelan *digital rele over current and short circuit high impedance* :

Penelitian ini bertujuan untuk permodelan perancangan sistem koordinasi pada *over current and short circuit high impedance* , dimana proses pengaturan dan pemantauan kondisi tiap rele dapat di lakukan satu lokasi yang sama ,sehingga memepermudah operator dalam melakukan perubahan dan pemantauan

1.5 Metodologi Penelitian

Dalam pembuatan tugas akhir ini yang berjudul "Desain Prototipe Dan Koordinasi Digital Relay Satu Fasa Terhadap Gangguan Overcurrent" penulis melakukan beberapa tahapan kegiatan dalam proses pembuatan tugas akhir ini, adapun tahapan yang di lakukan yaitu tahap pemilihan judul tugas akhir (studi literatur), tahap perencanaan dan pengerjaan alat, tahap pengambilan dan Analisa alat, dan yang terakhir yaitu pembuatan buku laporan.

Pada tahap perencanaan dan permodelan alat (studi literatur) ini penulis mempelajari tentang konsep sistem distribusi tenaga listrik dan peralatan yang terdapat pada sistem tersebut , setelah itu penulis mencari referensi yang berhubungan dengan konsep yang di ambil,beberapa referensi yang di perlukan yaitu pembacaan sensor arus dan tegangan pada sumber listrik AC, sistematika proteksi pada tegangan listrik satu fasa, dan mempelajari mengenai konsep kerja dan koordinasi OCR(*Over Current Relay)*.

Untuk tahap perencanaan dan pengerjaan terdiri dari *hardware* dan *software* yang meliputi perancangan *prototype* relai dilengkapi dengan sensor arus, sensor tegangan, relai DC, komunikasi RS485 dan kontaktor yang nantinya akan dikendalikan degan mikrokontroler (Arduino Mega 2560). Pada tahap ini akan dilakukan pembuatan program pada Arduino untuk pembacaan sensor arus dan tegangan yang nantinya hasil dari pembacaan sensor tersebut akan mengendalikan relai DC untuk memicu *coil* kontaktor untuk membuka dan menutup.

Setelah itu dilakukan pengujian alat, di mana sensor dan komponen di kalibrasikan dengan peralatan yang telah terkalibrasi, setelah mendapatkan beberapa data maka proses berikutnya yaitu menganalisa kesalahan atau kegagalan pada alat dan mengatasi permasalahan tersebut. Tahapan ini dilakukan dengan melakukan pengujian kerja setiap relai dan koordinasi antar relai. Tahap setelah alat di nyatakan sesuai maka tahap akhir penelitian adalah penyusunan laporan penelitian yang menjelaskan semua tahap – tahap sebelumnya yang nantinya akan di paparkan sebagai tugas akhir.

1.6 Sistematika Laporan

Sistematika pembahasan Tugas Akhir ini terdiri dari lima bab, yaitu pendahuluan, teori penunjang, perencanaan dan pembuatan alat, pengujian dan analisa alat, serta penutup.

BAB I PENDAHULUAN

Membahas tentang latar belakang, permasalahan,batasan masalah, maksud dan tujuan, sistematika laporan, serta relevansi.

BAB II TEORI PENUNJANG

Berisi teori penunjang yang mendukung dalam perencanaan dan pembuatan alat.

BAB III PERANCANGAN ALAT

Membahas tentang perencanaan dan pembuatan perangkat keras yang meliputi rangkaian-rangkaian, desain bangun, dan perangkat lunak yang meliputi program yang akan digunakan untuk mengaktifkan alat tersebut.

BAB IV PENGUJIAN DAN ANALISA ALAT

Membahas tentang pengukuran, pengujian, dan penganalisaan terhadap kepresisian sensor dan alat yang telah dibuat.

BAB V PENUTUP

Menjelaskan tentang kesimpulan dari Tugas Akhir ini dan saran-saran untuk pengembangan alat ini lebih lanjut.

1.7 Relevansi

Diharapkan alat ini dapat terealisasi, alat ini dapat digunakan untuk Untuk mempermudah proses pembelajaran dalam koordinasi *setting Over Current Relay dan short circuit*.

BAB II
TEORI DASAR

Pada Bab II ini akan dijelaskan mengenai teori-teori dasar yang menunjang dan berhubungan dalam pengerjaan Tugas Akhir ini. Teori dasar ini diharapkan mampu membantu dalam pengerjaan Tugas Akhir dan dapat dijadikan referensi nantinya.

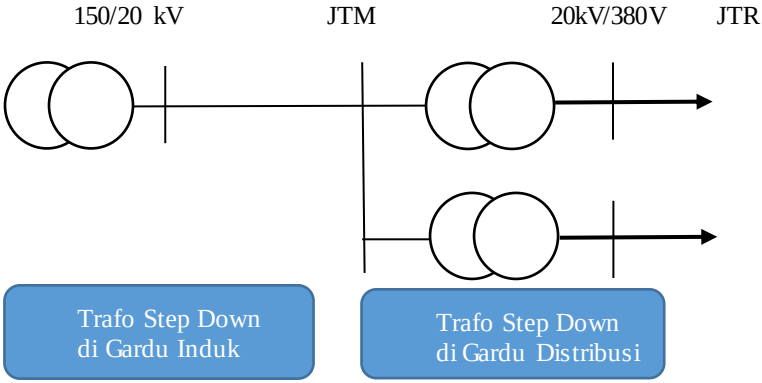
2.1 Jaringan distribusi tegangan menengah

Sistem distribusi tenaga listrik merupakan salah satu bagian dari suatu sistem tenaga listrik yang dimulai dari PMT incoming di gardu induk sampai dengan Alat Penghitung dan Pembatas (APP) di instalasi konsumen yang berfungsi untuk menyalurkan dan mendistribusikan tenaga listrik dari gardu induk sebagai pusat pusat beban ke pelanggan pelanggan secara langsung atau melalui gardu-gardu distribusi (gardu trafo) dengan mutu yang memadai sesuai standar pelayanan yang berlaku.

Dilihat dari tegangannya sistem distribusi pada saat ini dapat dibedakan dalam 2 macam yaitu

a. Distribusi Primer, sering disebut sistem Jaringan Tegangan Menengah (JTM) dengan tegangan operasi nominal 20 kV/ 11,6 kV

b. Distribusi Sekunder, sering disebut sistem Jaringan Tegangan Rendah (JTR) dengan tegangan operasi nominal 380 / 220 Volt



Gambar 2.1 **Sistem Jaringan Distribusi**

Jaringan distribusi primer (Jaringan Distribusi Tegangan Menengah) merupakan suatu jaringan yang letaknya sebelum gardu ditribusi yang di tunjukkan pada Gambar 2.1, berfungsi menyalurkan tenaga listrik bertegangan menengah 20 kV. Hantaran dapat berupa kabel dalam tanah atau saluran/kawat udara yang menghubungkan gardu induk (sekunder trafo) dengan gardu distribusi atau gardu hubung (sisi primer trafo distribusi).

2.2 Arus gangguan hubung singkat

Dalam operasi sistem tenaga listrik sering terjadi gangguan – gangguan yang dapat mengakibatkan terganggunya penyaluran tenaga listrik kekonsumen. Gangguan adalah penghalang dari suatu sistem yang sedang beroperasi atau suatu keadaan dari sistem penyaluran tenaga listr ik yang menyimpang dari kondisi normal. Berdasarkan ANSI/IEEE Std.100-1992 gangguan didefenisikan sebagai suatu kondisi fisis yang disebabkan kegagalan suatu perangkat, komponen atau suatu elemenuntuk bekerja sesuai dengan fungsinya. Gangguan hampir selalu ditimbulkan oleh hubung singkat antar fasa atau hubung singkat fasa ke tanah. Suatu gangguan hampir selalu berupa hubung langsung atau melalui impedansi. Istilah gangguan identik dengan hubung singkat, sesuai standar ANSI/IEEE Std. 100-1992. Sesuai dengan SPLN No 2:1978 Pasal 9 bahwa dalam sistem distribusi listrik tiga fasa tiga kawat menggunakan pentanahan dengan tahanan sebagai suatu sistem distribusi yang berlaku untuk wilayah kerja PLN di seluruh Indonesia. Dan dalam Pasal 10 dijelaskan bahwa tanahan operasi tersebut terdiri dari tahanan rendah dan tahanan tinggi , untuk sistem distribusi 20 kV dengan tahanan rendah terutama wilayah Jawa Barat dan Jakarta Raya sedangakan sistem distribusi 20 kV dengan tahanan tinggi terdapat di Jawa Timur. Untuk sistem Jawa Timur yang menggunakan pentanahan dengan tahanan tingi maka nilai tahanan yang digunakan adalah 500Ω dengan arus gangguan hubung singkat satu fasa ke tanah maksimal 25A jika mengunakan saluran udara. Pada sistem ini pula arus hubung singkat 2 fasa biasanya lebih kecil dari pada arus hubung singkat 3 fasa. Sedang arus gangguan 1 fasa ketanah hampir selalu lebih kecil daripada arus hubung singkat 3 fasa karena: 1. Umumnya impedansi urutan nolnya lebih besar dari pada impedansi urutan positif/ negatif. 2. Gangguan tanah melalui tahanan gangguan 3. Untuk pentanahan yang menggunakan tahanan, tahanan netralnya akan membatasi arus gangguan 1 fasa ketanah. Kondisi

sebaliknya terjadi (gangguan 1 fasa ke tanah lebih besar dari pada arus hubung singkat 3 fasa) apabila lokasi gangguan berada di pusat pembangkit atau dekat pusat pembangkit pada sistem dengan pentanahan langsung (solid grounded).

2.2.1 Penyebab Gangguan Hubung Singkat

Gangguan biasanya diakibatkan oleh kegagalan isolasi di antara penghantar fasa atau antara penghantar fasa dangan tanah. Secara nyata kegagalan isolasi dapat menghasilkan efek pada sistem yaitu menghasilkan arus yang cukup besar. Ada beberapa faktor yang menyebabkan terjadinya gangguan, antara lain sebagai berikut :

a. Faktor Manusia

   Faktor ini terutama menyangkut kesalahan atau kelalaian dalam memberikan perlakuan pada sistem. Misalnya salah menyambungkan rangkaian, salah dalam mengkalibrasi suatu piranti pengaman, dan sebagainya.

b. Faktor Internal

   Faktor ini menyangkut gangguan – gangguan yang berasal dari sistem itu sendiri. Misalnya usia pakai, keausan, dan sebagainya. Hal ini bisa mengurangi sensitivitas relai pengaman, juga mengurangi daya isolasi peralatan listrik lainnya.

c. Faktor Eksternal

   Faktor ini meliputi gangguan – gangguan yang berasal dari lingkungan di sekitar sistem misalnya cuaca, gempa bumi, banjir, dan sambaran petir. Disamping itu ada kemungkinan gangguan dari binatang, misalnya gigitan tikus, burung, kelelawar, ular dan sebagainya. Gangguan-gangguan tersebut menyebabkan terjadinya:

   1. Interupsi kontinuitas pelayanan daya kepada para konsumen apabila gangguan itu sampai menyebabkan terputusnya suatu rangkaian (circuit) atau menyebabkan keluarnya satu unit pembangkit.

   2. Penurunan tegangan yang cukup besar menyebabkan rendahnya kualitas tenaga listrik dan merintangi kerja normal pada peralatan konsumen.

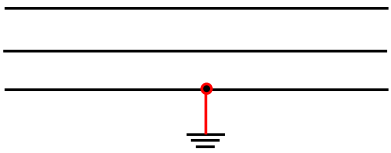
3. Pengurangan stabilitas sistem dan menyebabkan jatuhnya generator. Dan merusak peralatan pada daerah terjadinya gangguan itu.

2.2.2 Jenis Jenis Gangguan Hubung Singkat

Pada dasarnya gangguan yang sering terjadi pada sistem distribusi saluran 20 kV dapat digolongkan menjadi dua macam yaitu gangguan dari dalam sistem dan gangguan dari luar sistem. Gangguan yang berasal dari luar sistem disebabkan oleh sentuhan daun/pohon pada penghantar, sambaran petir, manusia, binatang, cuaca dan lain-lain. Sedangkan gangguan yang datang dari dalam sistem dapat berupa kegagalan dari fungsi peralatan jaringan, kerusakan dari peralatan jaringan, kerusakan dari peralatan pemutus beban dan kesalahan pada alat pendeteksi.

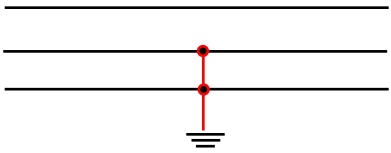
Klasifikasi gangguan yang terjadi pada jaringan distribusi sebagai berikut, seperti pada Gambar 2.2 sampai Gambar 2.6.

1) Gangguan hubung singkat satu fasa ke tanah



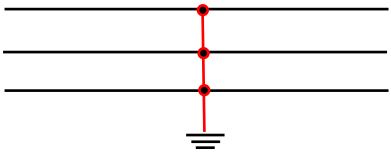
Gambar 2.2 **Gangguan Satu Fasa Ke Tanah**

2) Gangguan hubung singkat dua fasa ke tanah



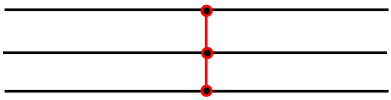
Gambar 2.3 **Gangguan Dua Fasa Ke Tanah**

3) Gangguan hubung singkat tiga fasa ke tanah



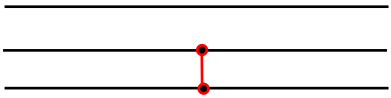
Gambar 2.4 **Gangguan Tiga Fasa Ke Tanah**

4) Gangguan hubung singkat tiga fasa



Gambar 2.5 **Gangguan Tiga Fasa**

5) Hubung singkat dua fasa



Gambar 2.6 **Gangguan Dua Fasa**

2.3 RELAY PROTEKSI

Rele adalah alat yang memproteksi sistem tenaga listrik dengan cara mendeteksi gangguan yang terjadi pada saluran, jika terjadi gangguan maka rele akan memberikan perintah circuit breaker membuka rangkaian untuk memutuskan arus yang menyebabkan gangguan tersebut.

Klasifikasi Rele Berdasarkan Besaran Input:

a Arus [I] : Rele Arus lebih (OCR), Rele Arus kurang (UCR)

b Tegangan [V] : Rele tegangan lebih [OVR], Rele tegangan kurang [UVR]

c Frekuensi [f] : Rele frekuensi lebih {OFR], Rele frekuensi kurang [UFR]

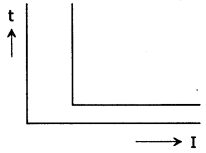
d Daya [P;Q] : Rele daya Max/Min, Rele arah/ Directional, Rele Daya balik.

e Impedansi [Z] : Rele jarak [Distance]

f Beda arus : Rele diferensial

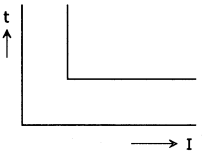
Berdasarkan Karakteristik Waktu Kerja:

a Seketika [Rele instant / Moment /high speed]

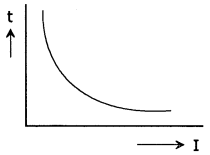


b Penundaan waktu [time delay]

1. Definite time rele



2. Inverse time rele



Fungsi Rele:

a Secara umum rele berfungsi memberikan instruksi kepada rangkaian pemutus (circuit breaker/CB) untuk mengisolasi sistem yang mengalami gangguan.

b Secara khusus, fungsi masing – masing rele tergantung kepada karakteristik dan besaran input yang mempengaruhi kerja rele misalnya:

1. Rele arus lebih (Over Current Rele/OCR) berfungsi melindungi systemdari gangguan arus lebih.
2. Rele impedansi berfungsi melindungi sistem dari gangguan yang terkait dengan perubahan impedansi saluran.

3. Rele jarak berfungsi melindungi sistem dari gangguan berdasarkan besaran jarak tertentu yang di *setting* pada rele.

2.3.1 Koordinasi Setting Over Current Relay

Aplikasi perhitungan dahulu digunakan untuk menentukan pengaturan yang sesuai pada rele konvensional, namun untuk rele saat ini sudah banyak yang memakai rele digital dimana pengaturan dapat diatur dan dikontrol dengan *microcontroller*. Tetapi perhitungan pengaturan rele OCR tetap dilakukan sebagai ilmu dasar untuk melakukan teknik men-*setting* suatu rele.

Berikut merupakan rumus waktu kerja rele yang dipakai :

$$t = \frac{\alpha \; x \; tms}{\{(\frac{If}{Iset})^{\beta} - 1\}}$$

Keterangan :

If = Arus gangguan

Iset = Arus *setting*

tms = time dial

α = koefisien kali

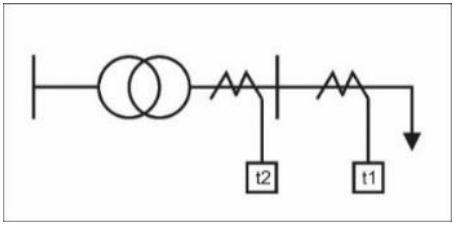
β = koefisien pangkat

$$tms = \frac{t \; x \; \{(\frac{If}{Iset})^{\beta} - 1\}}{\alpha}$$

Tabel 2.1 **Karakteristik Relay**

KARAKTERISTIK RELAY	α	β
STANDARD INVERSE	0.14	0.02
LONG TIME INVERSE	120	1
EXTREMELY INVERSE	80	2
VERY INVERSE	13.5	1
SHORT TIME INVERSE	0.05	0.04

Tabel 2.1 menunjukkan nilai α dan β dari setiap karakteristik relay. Sedangkan untuk rele yang penulis hitung adalah rele pada sisi sekunder yaitu Rele Arus Lebih pada penyulang dan PMCB. Untuk gambar diagram satu garis koordinasi rele dapat dilihat pada gambar dibawah ini.



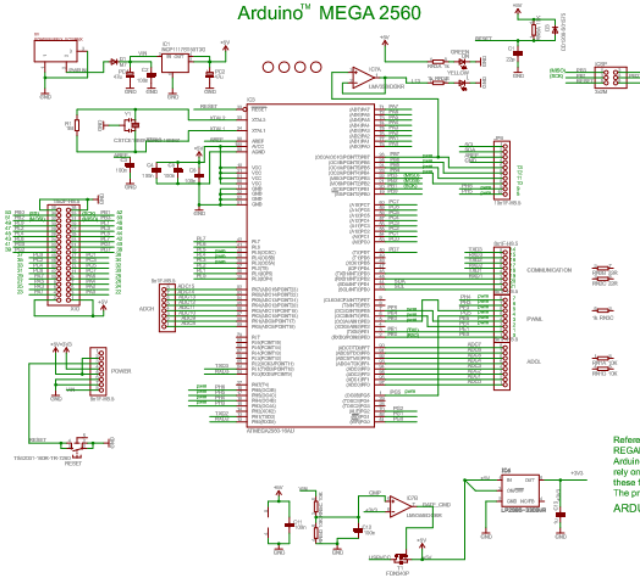
Gambar 2.7 **Sistem Koordinasi *Over Current Relay***

Gambar 2.7 diatas menunjukkan diagram *wiring* dari system koordinasi relay. Sebagai catatan bahwa nilai waktu kerja yang paling

cepat adalah rele yang paling jauh dari sumber, ini berarti waktu kerja t1 lebih cepat dibanding t2.

2.4 Arduino Mega 2560

Arduino Mega 2560 adalah papan mikrokontroler berdasarkan ATmega2560. Arduino ini memiliki 54 digital pin input / output (yang 15 dapat digunakan sebagai output PWM), 16 analog input, 4 UART (hardware port serial), 16 MHz osilator kristal, koneksi USB, jack listrik, header ICSP, dan tombol reset. Arduino ini berisi semua yang diperlukan untuk mendukung mikrokontroler; untuk dapat terhubung ke komputer dengan meggunakan kabel USB atau sumber tegangan berasal dari adaptor AC-DC atau baterai untuk menghidupkan Arduino. Bentuk fisik Arduino mega 2560 seperti gambar 2.8



Gambar 2.8 ***Board* Arduino Mega 2560**

Spesifikasi Arduino Mega 2560 adalah sebagai berikut:

1. Menggunakan chip microcontroller AtMega2560.
2. Tegangan operasi 5 Volt.
3. Tegangan input (yang direkomendasikan, via jack DC) sebesar 7-12 Volt.
4. Tegangan input batas sebesar 6-20V
5. Digital I/O sebanyak 54 buah, 6 diantaranya menyediakan PWM output.
6. Analog input pin sebanyak 16 buah.
7. Arus DC per pin I/O sebesar 20 mA.
8. Arus DC pada pin 3,3 Volt sebesar 50 mA.
9. Flash memory sebesar 256 KB, 8 KB telah digunakan untuk bootloader.
10. SRAM sebesar 8 kb.
11. EEPROM sebesar 4 kb.
12. Clock speed sebesar 16 Mhz.
13. Dimensi Arduino Mega 2560 sebesar 101,5 mm x 53,4 mm. 13. Berat Arduino Mega 2560 sebesar 37 g.

2.5 Sensor Arus SCT013

Transformator arus (CT) adalah sensor yang mengukur arus bolak-balik. Trafo SCT013 merupakan series travo pengukuran dengan jenis inti split, karena dapat dijepitkan langsung ke kabel bertegangan atau netral yang masuk pada beban yang akan diukur. Seperti trafo lainnya, transformator ini memiliki gulungan primer, inti magnetik, dan gulungan sekunder. Untuk monitoring, gulungan primer digunakan untuk masuknya kabel bertegangan atau netral (hanya salah satu). Dan gulungan sekundernya disambungan dengan alat pengukur (seperti rangkaian *microcontroller*). Bentuk ukuran yang kecil seperti gambar 2.9 d fungsikan sebagai sensor arus mulai dari 10 A sampai dengan 100A

Gambar 2.9 **Sensor Arus SCT013**

Arus bolak-balik yang mengalir di primer menghasilkan medan magnet di inti, yang menginduksi arus di belitan sekunder. Arus dalam gulungan sekunder sebanding dengan arus yang mengalir dalam gulungan primer.

2.6 Travo step down 350 ma

Travo step down 350 ma di gunakan sebagai sensor tegangan pada rele. Sensor tegangan sendiri merupakan alat yang digunakan untuk mendeteksi besar tegangan yang melalui suatu peralatan listrik. Sensor tegangan menggunakan trafo step down dengan rangkaian penyearah. Penggunaan trafo step down yaitu agar dapat menurunkan tegangan pada sisi primer terhadap sisi sekunder, sehingga pada sisi sekunder dapat digunakan untuk peralatan pengukuran. Prinsip kerja dari sebuah trafo pada gambar 2.10 adalah ketika kumparan primer dihubungkan dengan sumber tegangan bolakbalik, perubahan arus listrik pada kumparan primer menimbulkan medan magnet yang berubah.

Gambar 2.10 **Travo 350ma**

Medan magnet yang berubah diperkuat oleh adanya inti besi dan dihantarkan inti besi ke kumparan sekunder, sehingga pada ujung-ujung kumparan sekunder akan timbul ggl induksi. Jika efisiensi sempurna, semua daya pada lilitan primer akan dilimpahkan ke lilitan sekunder.

.

2.6 *KONTAKTOR SATU FASA*

Sebuah kontaktor biasanya di pergunakan untuk tujuan proteksi maupun tujuan otomasi. Dalam hal otomasi ,sebuah rangkaian kontaktor biasanya di hubungkan terlebuh dahulu dengan sebuah alat control , counter ataupun timer , sedangkan untuk tujuan keamanan , sebuah kontaktor dapat saja di hubungkan dengan alarm , skring , sensor ataupun peralatan lainya , dalam proses kinerjanya sinyal dari alat kontroler akan memiliki amper yang kecil sehingga di perlukan suatu rangkaian tambahan sebagai actuator , sehingga arus yang mengalir pada kontaktor bernilai besar sehngga kontaktor dapat bekerja.

Pada kontaktor satu fasa gambar 2.11 nilai arus yang dapat di lewati tidaklah terlalu besar , berdasarkan nominal yang tertera pada kontaktor , kontaktor tersebut dapat di aliri beban motor sebesar 9 Ampere sedang kan untuk beban normal dan induktif rendah sebesar 25 Ampere.

Gambar 2.11 ***Kontaktor Satu Fasa***

2.7 Switch-Mode Power Supply

Switch-Mode Power Supply (SMPS) adalah jenis Power Supply yang langsung menyearahkan (rectify) dan menyaring (filter) tegangan Input AC untuk mendapatkan tegangan DC. Tegangan DC tersebut kemudian di-switch ON dan OFF pada frekuensi tinggi dengan sirkuit frekuensi

tinggi sehingga menghasilkan arus AC yang dapat melewati Transformator Frekuensi Tinggi. Tegangan yang dapat di keluarkan oleh switch mode power supply yang di gunakan adalah 12 volt dengan arus maksimal sebesar 2 Ampere. Switch-mode Power Supply yang di gunakan seperti pada gambar 2.12

Gambar 2.12 **Switch-Mode Power Supply**

2.8 *Uninterruptible Power Supply* (UPS) Inforce 650 VA

UPS adalah singkatan dari *uninterruptible power supply* sebagai alat *back-up* listrik ketika suatu peralatan elektronik kehilangan energi dari sumber utamanya. UPS pada gambar 2.13 bekerja diantara *device* yang di suplai dan colokan listrik, dari colokan listrik yang di alirkan ke baterai yang berada pada UPS dan kemudian di simpan untuk kestabilan tegangan energi. Listrik yang di simpan pada baterai akan di pakai ketika sumber energi utama listrik terputus.

Gambar 2.13 ***Uninterruptible Power Supply*** **(UPS) Inforce 650 VA**

Uninterruptible Power Supply (UPS) Inforce 650 VA memiliki spesifikasi sebagai berikut :

1. Kapasitas daya 650 VA
2. *Range* input 140V – 300VAC
3. Stabilitas *output* 230 V + /- 10%
4. 3 *Step* AVR
5. Efisiensi tinggi pada mode *in-line* 95%
6. Tipe baterai 8.2 Ah
7. 2 soket output
8. Indikator LCD

-----Halaman ini sengaja dikosongkan-----

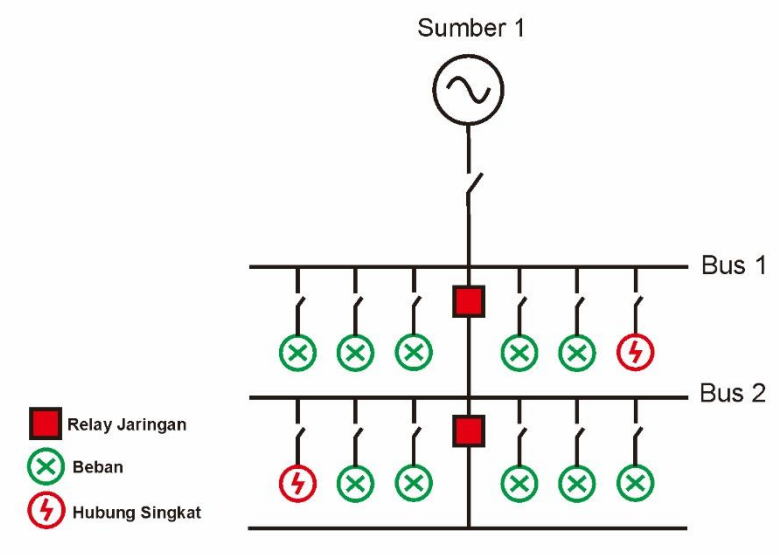
BAB III
PERENCANAAN ALAT

Pada bab 3 ini membahas mengenai perencanaan alat yang terdiri dari perancangan serta pembuatan alat pada Tugas Akhir kami yang berjudul **"DESAIN PROTOTIPE DAN KOORDINASI DIGITAL RELAY SATU FASA TERHADAP GANGGUAN OVERCURRENT".** Dalam perancangan dan pembuatan alat ini dibagi menjadi dua yaitu pada perangkat elektronik (*hadware*) dan pada perangkat lunak (*software*). Untuk bagian saya meliputi:

1. Perancangan *Hardware* terdiri dari :
   a Perancangan Sensor Tegangan
   b Perancangan Sensor Arus
   c Perancangan Relay Bantu dan MCB
   d Perancangan Shield Arduino Mega
2. Perancangan *Software* yang berupa *flowchart* terdiri dari :
   a Pemrograman Arduino IDE
   b Pemrograman Sensor Tegangan
   c Pemrograman Sensor Arus
   d Pemrograman Relay Bantu dan MCB

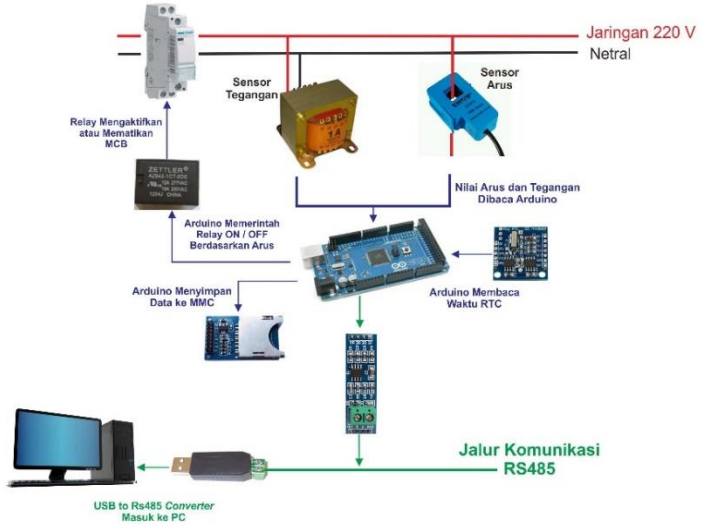
3.1 Diagram Fungsional Alat

Perencanaan Tugas Akhir "Desain Prototipe Dan Koordinasi Digital Relay Satu Fasa Terhadap Gangguan Overcurrent" ini mengenai sistem kerja alat secara keseluruhan. Dimana pada alat ini berfungsi sebagai pengaturan dan *monitor* keadaan arus dan tegangan jaringan dan mengamankan dari adanya arus lebih yang disebabkan hubung singkat dan beban lebih.Permodelan saluran yang digunakan pada alat ini seprti yang ditunjukkan pada gambar 3.1. Koordinasi relai dalam sistem memiliki arti bahwa dalam pemasangan relai harus memiliki koordinasi waktu agar nantinya dapat mengamankan jaringan yang memiliki gangguan dan tidak mengganggu jaringan lainnya.



Gambar 3.1 **Skema Sistem Secara Keseluruhan**

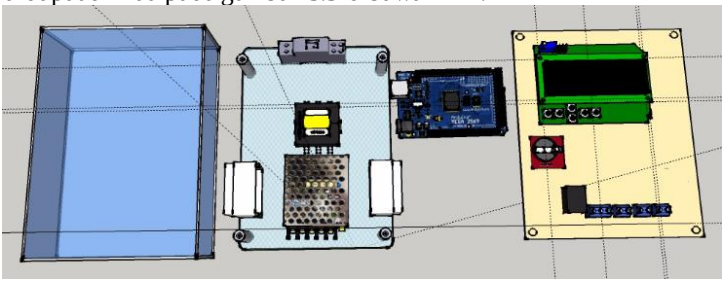
Sistem kerja dari relai ini dimulai dari memberikan *setting* pada relai yang terdiri dari *setting* arus, waktu, dan tipe karakteristik yang akan digunakan. Selanjutnya mengambil nilai dari sensor arus dan sensor tegangan, setelah terdeteksi bahwa arus pada jaringan adalah arus nominal, maka MCB akan *close* dan menyambungkan jaringan. Tetapi jika sensor membaca nilai dari arus nominal lebih kecil daripada arus *setting* maka MCB akan trip sesuai dengan waktu dan tipe karakteristik yang digunakan. Selanjutnya RTC akan membaca waktu dan tanggal secara *real time*, ini digunakan untuk pendataan kondisi dari jaringan agar dapat dipantau *user* secara *real time*. LCD pada relai akan menampilkan tegangan dan arus saat ini yang terukur dari jaringan. Kondisi tersebut akan disimpan di data *logger* dan akan dikirimkan ke PC untuk memonitor secara langsung. Pada PC ini dengan menggunakan software Lazarus akan menampilkan nilai arus nominal dan juga *setting* dari relai yang digunakan dan juga bisa memberi *setting* lewat Lazarus jika perlu. Permodelan perancangan alat seperti yang di tunjukkan pada gambar 3.2



Gambar 3.2 **Skema Relai**

3.2 Perancangan Elektronik

Pada perancangan elektronik (*hardware*) ini komponennya meliputi Rangkaian Sensor Tegangan, Rangkaian Sensor Arus, LCD Keypad 16 x 2, Komunikasi RS485, Relay dan MCB, Rangkaian RTC, Rangkaian SD Card, Shield Arduino Mega. Tampilan perancangan *hardware* untuk relai utama dapat dilihat pada gambar 3.3 dibawah ini :



Gambar 3.3 **Perancangan Elektronik Relai**

Susunan dari perancangan elektronik terbagi menjadi 2 bagian, yaitu, bagian atas dan bagian bawah. Pada bagian bawah terdapat *Power Supply*,

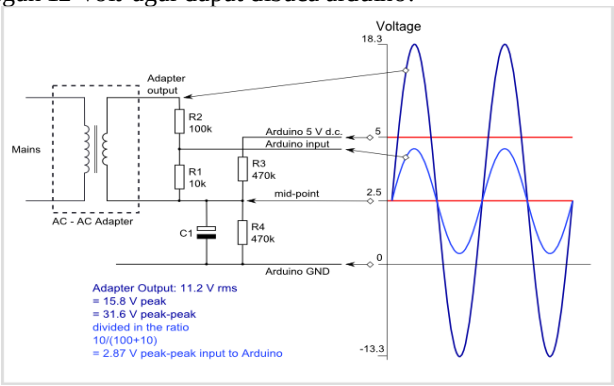
CT Sensor, Trafo sebagai sensor tegangan, MCB dan *Stop Contact* untuk menghubungkan jaringan dengan MCB yang ada pada relai. Sedangkan pada bagian atas terdapat 2 sisi bagian, pada sisi bawah terdapat *board* Arduino sebagai kontrol pada Tugas Akhir ini, sedangkan pada sisi atas terdapat rangkaian Sensor Tegangan, rangkaian Sensor Arus, rangkaian Relai penggerak MCB, LCD *Keypad* 16 x 2, Komunikasi RS485, RTC dan *SD Card*.

Tata letak dari komponen - komponen sistem kontrol ini dibuat seperti demikian untuk mempermudah dalam pengoperasiaannya. Selain itu dengan diletakkan menjadi satu *board* seperti Gambar 3.1 dan Gambar 3.2 diatas akan menjadi lebih rapi dan efisien. Mengenai komponen – komponen yang digunakan akan dijelaskan pada sub bab berikut ini.

3.2.1 Perancangan Sensor Tegangan

Tugas akhir ini menggunakan sensor tegangan dari *transformer step down* 220 Volt / 12 Volt. Trafo *stepdown* digunakan untuk menurunkan tegangan 220 Volt agar nantinya dapat dibaca oleh ADC pada arduino yang membutuhkan tegangan 0-5 Volt.

Agar hasil dari sensor tegangan dapat dibaca oleh ADC arduino maka dilakukan pembagian tegangan agar didapat nilai tegangan yang lebih kecil. Tegangan yang akan masuk pada ADC masih berupa gelombang sinus yang memiliki nilai negatif. Agar semua data dapat dibaca oleh arduino, nilai negatif pada tegangan harus diubah mejadi positif. Berikut adalah gambar 3.4 rangkaian untuk mengkonversi tegangan 12 Volt agar dapat dibaca arduino:



Gambar 3.4 **Rangkaian Sensor Tegangan**

Agar dapat dibaca ADC arduino, tegangan 12 Volt dari output trafo ditransformasikan dengan cara pembagian tegangan seperti berikut:

$$\textbf{Vout} = \frac{R1}{R1+R2} \textbf{ x Vin} \quad \text{.....(3.1)}$$

$$\text{Vout} = \frac{10K}{10\,K + 100K} \text{ x 12 Volt}$$

Vout = 1,09 Volt

Untuk mengubah tegangan negatif menjadi tegangan positif dengan menggunakan rangkaian clamper yang akan menggeser sinyal. Pertama dengan membuat tegangan *mid-point* dengan menggunakan 2 resistor yang sama besar:

$$\textbf{Vmid} = \frac{R1}{R1+R2} \textbf{ x Vin mid} \quad \text{.....(3.2)}$$

$$\text{Vmid} = \frac{470k}{470k + 470k} \text{ x 5 Volt}$$

Vmid = 2,5 Volt

Dengan begitu dapat disimpulkan bahwa tegangan yang masuk ke ADC Arduino Mega 2560 adalah sebagai berikut:

$$\textbf{V } \textit{pick to pick} \textbf{ = (2 x Vout) + Vmid} \quad \text{.....(3.3)}$$

V *pick to pick* = (2 x 1,09) + 2,5 Volt

V *pick to pick* = 4,68 Volt

$$\textbf{V pmax = Vmid + Vout} \quad \text{.....(3.4)}$$

V pmax = 2,5 Volt + 1,09 Volt

V pmax = 3,59 Volt

$$\textbf{Vpmin = Vmid – Vout} \quad \text{.....(3.5)}$$

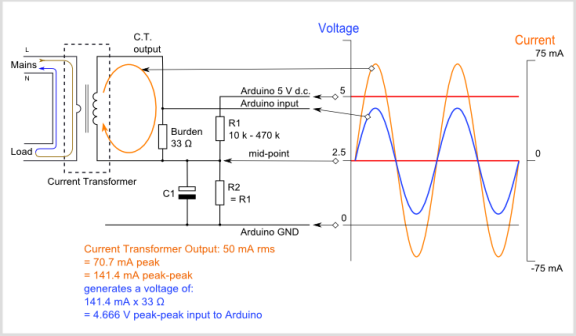
V pmin = 2,5 Volt – 1,09 Volt

V pmin = 1,41 Volt

3.2.2 Perancangan Sensor Arus

Sensor arus yang digunakan pada Tugas Akhir ini adalah sensor SCT013-10. Sensor ini dapat membaca arus hingga 10 Ampere dan akan mengubahnya menjadi tegangan sebesar 1 Volt. Sensor SCT013-10 dapat mengubah output menjadi tegangan karena dalam rangkaian sensor sudah terdapat resistor *burden* sebesar 180 Ω. Sehingga arus sebesar 10 A saat melewati R *burden* akan berubah menjadi tegangan 1 Volt.

Tegangan 1 Volt dari output sensor masih memiliki tegangan negatif. Tegangan ini yang nantinya akan dimasukkan pada ADC arduino. Agar semua data dapat dibaca oleh arduino, nilai negatif pada tegangan harus diubah mejadi positif. Berikut adalah gambar rangkaian untuk mengkonversi tegangan agar dapat dibaca arduino:



Gambar 3.5 **Rangkaian Sensor Arus**

Dengan sensor SCT13-10 yang memiliki Rasio Transformasi sebesar 1800 dan R *burden* = 180Ω, dapat diketahui besarnya arus yang ditransformasikan sensor sebelum menjadi tegangan adalah sebagai berikut :

$$\textbf{I}output\ \textbf{CT} = \frac{I\ input}{rasio} \quad \text{.....(3.6)}$$

$$I output\ CT = \frac{10\ A}{1800}$$

$$I output\ CT = 0{,}0055\ A$$

Tegangan yang dihasilkan oleh sensor arus setelah melewati resistor *burden* adalah sebesar :

V CT = R burden x Arus CT.....(3.7)

V CT = 180Ω x 0,0055 A

V CT = 0,99 Volt (1 Volt)

Untuk mengubah tegangan negatif menjadi tegangan positif dengan menggunakan rangkaian clamper yang akan menggeser sinyal. Pertama dengan membuat tegangan *mid-point* dengan menggunakan 2 resistor yang sama besar :

$$\textbf{Vmid} = \frac{R1}{R1+R2}\ \textbf{x Vin mid} \quad \text{.....(3.8)}$$

$$Vmid = \frac{10K}{10\ K+10K} \text{ x 5 Volt}$$

Vmid = 2,5 Volt

Dengan begitu dapat disimpulkan bahwa tegangan yang masuk ke ADC Arduino Mega 2560 adalah sebagai berikut:

V *pick to pick* = (2 x Vout) + Vmid .....(3.9)

V *pick to pick* = (2 x 1) + 2,5 Volt

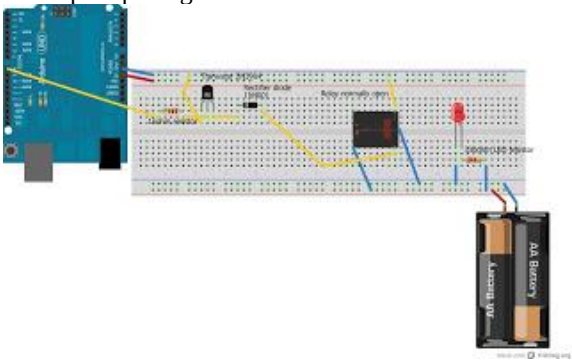
V *pick to pick* = 4,5 Volt

V pmax = Vmid + Vout .....(3.10)
V pmax = 2,5 Volt + 1 Volt
V pmax = 3,5 Volt
Vpmin = Vmid – Vout .....(3.11)
V pmin = 2,5 Volt – 1 Volt
V pmin = 1,5 Volt

3.2.3 Perancangan Modul Rangkaian RTC DS1307

Relay pada Tugas Akhir ini berguna untuk mengirimkan sinyal pada pemutus jaringan saat terdapat gangguan yang terbaca oleh arduino. Saat arduino membaca arus gangguan dari sensor CT, arduino akan mengaktifkan *relay* yang nantinya akan memerintah MCB untuk memutuskan saluran.

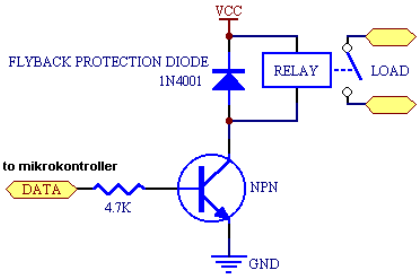
Relay akan dihubungkan pada pin ADC Arduino. Namun *relay* ini tidak mampu bekerja hanya dengan mengandalkan sinyal dari pin ADC, sehingga dipasanglah transistor sebagai *driver* relai. *Driver* ini berfungsi untuk menggerakkan *coil* yang ada pada *relay*. Rangkaian yang di gunakan seperti pada gambar 3.6



Gambar 3.6 **Rangkaian *Relay* dengan Arduino**

Transistor yang digunakan adalah transitor NPN. Basis dari transistor dihubungkan dengan resistor dan masuk ke ADC dari Arduino. Pin kolektor terhubung dengan dioda dan terhubung pada *coil relay*. Dioda disini berguna untuk mencegah arus balik terpengaruhnya arduino dari lonjakan arus yang terjadi ketika *coil relay* bekerja. Sedangkan pin

emitter terhubung dengan ground arduino. Gambar 3.7 merupakan skema rangkaian *relay* yang akan digunakan pada tugas akhir ini:



Gambar 3.7 **Skema Perancangan Rangkaian *Relay***

Modul RTC DS1307 dapat langsung dihubungkan pada arduino. Sambungan dengan Arduino Mega dapat dituliskan sebagai berikut :

1. VCC – pin 5V Arduino Mega
2. GND – pin *Ground* Arduino Mega
3. SDA – pin 20 (SDA) pada Arduino Mega
4. SCL – pin 21 (SDL) pada Arduino Mega

3.3 Perancangan Perangkat Lunak (*Software)*

Perancangan perangkat lunak (*software*) menggunakan dua macam *software* yakni Arduino IDE untuk pemrograman pada papan Arduino serta menjalankan fungsi dari sensor dan perangkat elektronik yang terhubung pada Arduino, sedangkan software Lazarus digunakan untuk merancang pada sisi *interface* dengan PC yang nantinya akan menampilkan dan memberikan *setting* pada relai menggunakan protokol Modbus RTU.

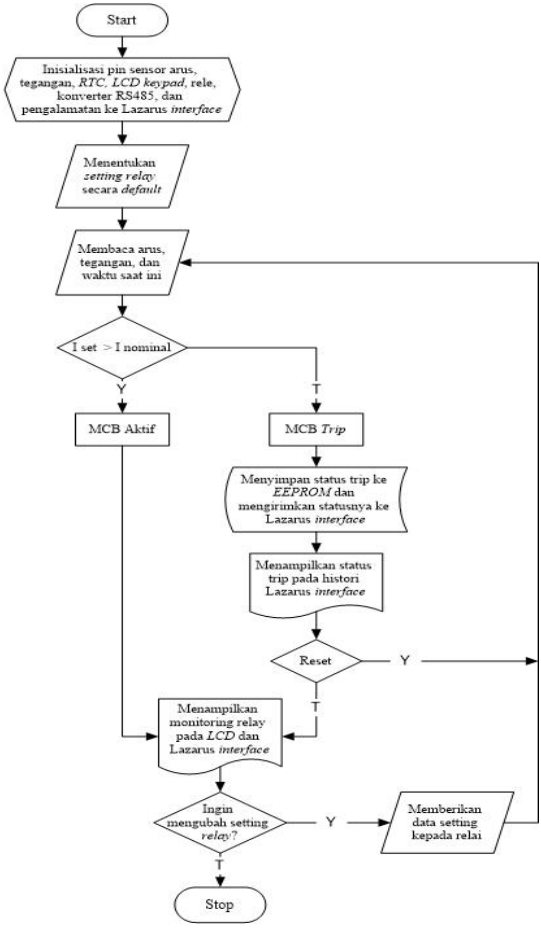
3.3.1 Pemrograman *Software* Arduino IDE

Software Arduino IDE pada Tugas Akhir digunakan untuk melakukan pemrograman papan Arduino dalam menjalankan sistem secara keseluruhan. Dan arduino yang kami gunakan adalah Arduino Mega. *Software* ini menggunakan bahasa pemrograman C. Perancangan dari pemrograman ini seperti yang ditunjukkan pada Gambar 3.8, dimana ketika program pertama kali dijalankan akan menyimpan *setting default* dari relai yang meliputi *setting* arus, waktu, dan tipe karakteristik relai yang digunakan. Selanjutnya relai akan membaca besar arus, besar tegangan yang terdapat dalam jaringan dan juga waktu yang dibaca oleh

RTC. Jika arus nominal jaringan kurang dari arus *setting*, maka MCB relai utama akan terhubung. Namun jika arus nominal lebih besar dari arus *setting* maka relai akan mendeteksi gangguan arus lebih dan MCB akan *trip* atau memutuskan jaringan sesuai dengan waktu dan tipe karakteristik relai yang digunakan.

Pada saat itu juga status trip pada relai akan disimpan di EEP ROM pada arduino mega yang kapasitasnya 4 kb. EEPROM ini adalah sejenis chip memori tidak-terhapus yang digunakan dalam komputer dan peralatan elektronik lain (seperti Arduino mega) untuk menyimpan sejumlah konfigurasi data pada alat elektronik tersebut yang tetap harus terjaga meskipun sumber daya diputuskan. Setelah disimpan, status trip akan dikirimkan ke *interface* lazarus sebagai data logger yang akan menampilkan histori trip dari relai. Tetapi apabila tombol reset relai ditekan maka status trip akan menjadi status normal kembali dan relai akan mulai membaca arus pada jaringan lagi. Data yang ada pada relai akan ditampilkan pada LCD, disimpan pada data logger dan dikirimkan ke lazarus sebagai *interface* di komputer. Unutk LCD akan menampilkan arus *setting*, arus nominal dan tegangan, waktu saat ini, dan juga status relai.

Dan untuk lazarus disini menampilkan arus *setting*, arus nominal, status relai, waktu *setting trip*, waktu *countdown trip*, waktu saat ini, dan juga tipe karakteristik yang digunakan pada kedua relai. Jika *setting* pada relai ingin diganti, maka bisa di ganti lewat lazarus ini, dengan cara mengubah *setting* yang diperlukan lalu dimasukkan ke dalam kolom lazarus dan dikirim ke relai yang perlu di *setting* ulang. Untuk lebih ringkasnya dapat dilihat pada gambar 3.8 yang merupakan *flowchart* pemrograman Arduino IDE dibawah ini :

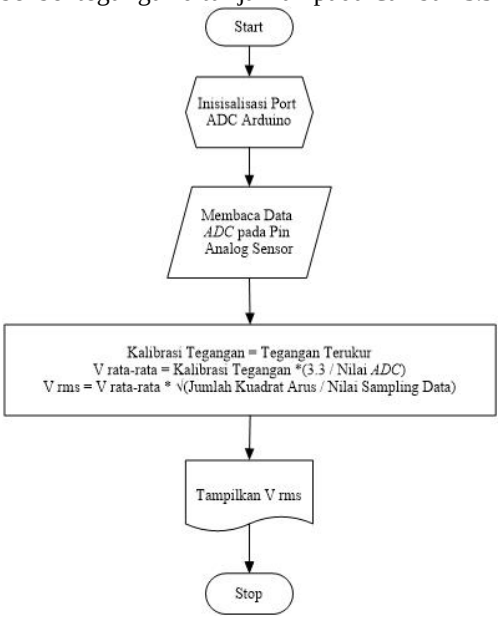


Gambar 3.8 ***Flowchart* Pemrograman *Softwarre* Arduino IDE**

3.3.2 Pemrograman Sensor Tegangan

Sensor tegangan yang digunakan pada Tugas Akhir ini menggunakan trafo CT 350 mA dengan tegangan 230 Volt / 9 Volt. Dimana *output* dari trafo ini berupa tegangan 9 Volt yang memiliki siklus negatif. Dengan konversi yang telah dijelaskan pada perancangan *hardware* tepatnya pada persamaan 3.1, 3.2 dan 3.3 maka akan didapat tegangan yang dapat terbaca oleh Arduino. Agar sensor tegangan ini dapat digunakan sebagai

sensor, dibutuhkan program yang sesuai untuk sensor ini. *Flowchart* untuk program sensor tegangan ditunjukkan pada Gambar 3.9



Gambar 3.9 ***Flowchart* Pemrograman Sensor Tegangan**

Seperti yang telah digambarkan pada *Flowchart* diatas, untuk urutan cara kerja dari *flowchart* adalah sebagai berikut :

i. *Start* adalah ketika program dimulai.
ii. Untuk *wiring* Sensor Tegangan dengan Arduino dihubungkan dengan pin analog Arduino atau pin ADC
iii. Selanjutnya inisialisasi variabel yang akan digunakan untuk pembacaan dan penghitungan tegangan.
iv. Selanjutnya membaca data pada analog Arduino atau ADC
v. Dengan menggunakan perhitungan sebagai berikut akan memperoleh nilai V rms :

Kalibrasi Tegangan = Tegangan Terukur .....(3.12)

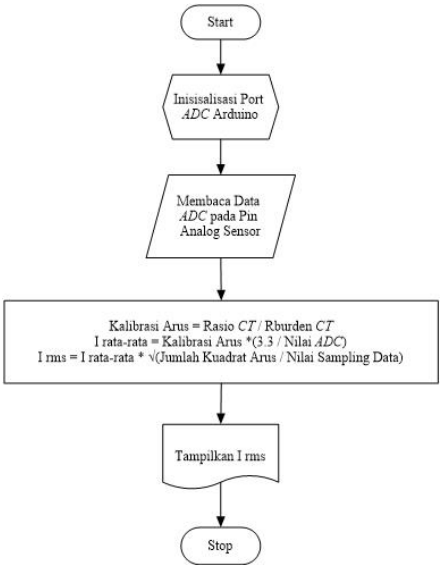
V rata-rata = Kalibrasi Tegangan $*\frac{3.3}{Nilai\ ADC}$ .....(3.13)

$$V\ rms = V\ rata\text{-}rata * \sqrt{\frac{Jumlah\ Kuadrat\ Arus}{Nilai\ Sampling\ Data}} \quad \text{..... (3.14)}$$

vi. Menampilkan nilai V rms

3.3.3 Pemrograman Sensor Arus

Sensor arus yang digunakan pada Tugas Akhir ini adalah sensor arus *current transformers* SCT013-10 dimana *output* dari sensor ini berupa tegangan dan yang nantinya harus dapat terbaca oleh Arduino. Agar sensor arus ini dapat digunakan sebagai sensor, dibutuhkan program yang sesuai untuk sensor ini. *Flowchart* untuk program sensor arus ditunjukkan pada Gambar 3.10



Gambar 3.10 ***Flowchart* Pemrograman Sensor Arus SCT-013**

Seperti yang telah digambarkan pada *Flowchart* diatas, untuk urutan cara kerja dari *flowchart* adalah sebagai berikut :

1. *Start* adalah ketika program dimulai.
2. Untuk *wiring* Sensor Arus dengan Arduino dihubungkan dengan pin analog Arduino
3. Selanjutnya inisialisasi variabel yang akan digunakan untuk pembacaan dan penghitungan arus.

4. Selanjutnya membaca data pada analog Arduino atau ADC
5. Dengan menggunakan perhitungan sebagai berikut akan memperoleh nilai I rms :

Kalibrasi Arus = Rasio CT / Rburden CT .....(3.15)

$$\text{I rata-rata} = \text{Kalibrasi Arus} * \frac{3.3}{Nilai\ ADC} \quad \text{.....(3.16)}$$

$$\text{I rms} = \text{I rata-rata} * \sqrt{\frac{Jumlah\ Kuadrat\ Arus}{Nilai\ Sampling\ Data}} \quad \text{.....(3.17)}$$

6. Menampilkan nilai I rms

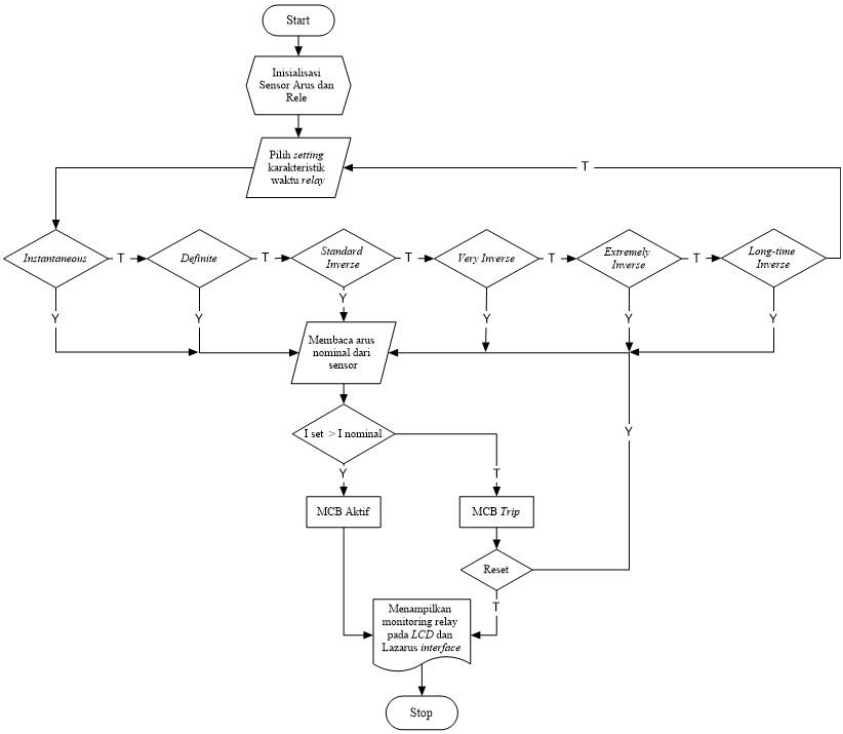
3.3.4 Pemrograman Relai

Relai ini berfungsi untuk mengamankan jaringan ketika sensor arus mendeteksi adanya arus lebih, sehingga relai ini memerintah MCB untuk memutuskan jaringan (*trip*). Selain itu relai ini juga dapat menampilkan status dari jaringan tersebut dan mengirimkannya ke data *logger* dan *human interface*. Agar relai dapat berjalan sesuai fungsinya, maka dibutuhkan program yang tepat pada papan Arduino.

Pertama, Arduino harus dapat mengenali SD *Card*, RTC, Modul RS485, Sensor Arus dan Sensor Tegangan. Selanjutnya adalah menentukan arus *setting* relai. Arus *setting* ini digunakan untuk membatasi besar arus yang diperbolehkan untuk melewati jaringan. *Setting* relai yang kami pakai ini memliki 6 tipe karakteristik yang memiliki waktu trip yang berbeda – beda. Lalu setelah mengenali sensor, Arduino akan membaca Sensor Arus, Tegangan dan Waktu RTC. Jika arus nominal atau arus yang terukur pada sensor kurang dari arus *setting* pada arduino, maka MCB pada relai akan aktif dan menghubungkan jaringan (antara sumber dan beban). Namun jika sebaliknya, jika arus nominal atau arus yang terukur pada jaringan lebih besar dari arus *setting* maka MCB akan *Trip* sesuai dengan tipe karakteristik yang dipilih. Besarnya arus ini merupakan arus gangguan, dapat berupa karena beban lebih (over load) atau juga dapat berupa arus hubung singkat (short circuit). Dengan *trip*-nya MCB ini, maka arus pada jaringan akan terputus sehingga antara sumber dan beban tidak terhubung. Atau dengan kata lain, jaringan atau sistem mati karena gangguan. Untuk mengembalikan agar jaringan tersambung lagi, maka perlu menekan tombol *reset* pada

relai agar Arduino dapat membaca ulang arus dan tegangan yang melewati jaringan.

Kondisi yang terjadi pada relai ini akan ditampilkan pada LCD yang ada pada relai, sehingga *user* dapat langsung mengecek keadaan relai saat ini. Selanjutnya adalah menyimpan kondisi relai pada sebuah SD *Card*. Penyimpanan data ini berguna untuk Data *Logger* pada relai. Data yang ada pada arduino selanjutnya dikirimkan pada Lazarus melalui komunikasi RS485. Yang setelah itu data akan diproses pada Lazarus dan menampilkan monitoring relai berupa nilai arus nominal dan nilai tegangan, waktu saat ini, nilai *setting* arus, nilai waktu *countdown trip*, nilai waktu *trip setting*, dan juga tipe karakteristik yang digunakan pada kedua relai. Gambar 3.11 berikut ini adalah *Flowchart* dari program Arduino.



Gambar 3.11 ***Flowchart*** **Pemrograman Relai Utama**

BAB IV
PENGUJIAN DAN ANALISA DATA

Dalam proses pembuatan alat di perlukan tahaban tahaban untuk memonitoring kinerja dari sistem yang di rancang Untuk mengetahui kinerja dari peralatan dan pembuatan sistem yang telah dirancang,maka diperlukan pengujian dan analisa data dari setiap komponen pendukung yang dibuat agar sistem dapat berjalan dengan baik sesuai dengan yang diharapkan.melakukan Pengujian pada suatu sistem secara terpisah merupakan salah satu langkah yang harus dilakukan untuk mengetahui apakah sistem yang telah dibuat sesuai dengan yang direncanakan, hal tersebut diharapakan apabila terjadi suatu kesalahan pada sistem maka dapat dengan cepat di ketahui bagian yang berkemungkinan besar terjadi masalah. Kesesuaian sistem dengan perencanaan dapat dilihat dari hasil-hasil yang dicapai pada pengujian sistem.

Bab ini berisi tentang pengujian dan analisa data *hardware* dan *software* yang telah dibuat. Adapun bagian – bagian yang akan diuji pada alat setelah di rancang adalah :

1. Pengujian UPS (*Uninterruptible Power Supply*)
2. Pengujian *Power Supply*
3. *Input/Output* Arduino
4. Pembacaan Sensor Arus
5. Pembacaan Sensor Tegangan
6. Pengujian Arus Impedansi Tinggi

Gambar 4.1 **Bentuk Fisik Prototipe Digital Relay**

4.1 Pengujian UPS (*Uninterruptible Power Supply*)

Pada dasarnya UPS merupakan cadangan sumber pada peraalatan elektronik , Pengujian UPS bertujuan untuk mengetahui besar tegangan pada UPS ketika dilakukan mode baterai dan saat diberi masukan 223 Volt. Pengujian ini dilakukan pada terminal *output* UPS. Pada Tugas Akhir ini, UPS dihubungkan pada *Power Supply* relai utama yang fungsinya sebagai sumber relai utama. Hal tersebut di maksudkan seperti layaknya batrai pada gardu induk yang memiliki nilai tegangan yang konstan ,fungsi batrai tersebut di gunakan sebagai sumber pada microcontroller sehingga nilai pembacaan data tidak terpengaruh oleh sumber jaringan yang tegangan dan frekuensinya berubah ubah setiap saat.

Pengukuran ini untuk mendapatkan besar arus, tegangan dan bentuk gelombang yang keluar dari UPS. Gambar 4.1 berikut ini adalah skema pengambilan data untuk UPS.

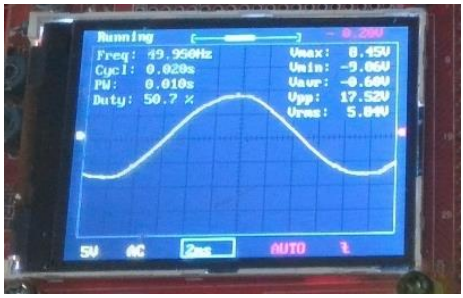
Gambar 4.2 **Skema Pengujian UPS**

Pengujian UPS ini menggunakan multimeter "SANWA CD800a" dilakukan pada 2 kondisi, yaitu ketika tidak menggunakan sumber AC (sebagai baterai) dan ketika menggunakan *input* sumber AC. Berikut adalah hasil pengujian UPS yang digunakan untuk *input* Power Supply dan dapat dilihat pada tabel 4.1

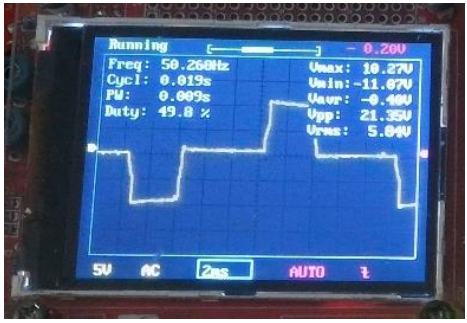
Tabel 4.1 **Pengambilan Data Tegangan Ups**

No	Mode	Hasil
1	Baterai	154,4 V
2	Dengan *Input* 223 Volt	223,4 V

Sebagai membuktikan bahwa keluaran UPS merupakan gelombang AC, maka dilakukan sebuah uji coba untuk melihat bentuk gelombang ups pada osiloskop. Sebelumnya, nilai tegangan *output* UPS dikecilkan terlebih dahulu dengan menghubungkan ke trafo *step-down* 220 / 5 Volt agar dapat dibaca oleh osiloskop. Berikut adalah bentuk gelombang dari output UPS yang ditunjukkan pada gambar 4.2 dan 4.3



Gambar 4.3 **Bentuk Gelombang UPS dengan *Input* 220 Volt**



Gambar 4.4 **Bentuk Gelombang UPS sebagai Baterai**

Gelombang yang dihasilkan UPS tanpa di hubungkan pada jaringan tidaklah murni gelombang AC hal tersebut di karenakan proses inverting dari UPS itu sendiri namun jika di konversi menjadi DC dengan tegangan 12 volt hal tersebut tidaklah masalah.

4.2 Pengujian *Power Supply*

Pengujian ini dilakukan pada *Power Supply* yang nantinya hasil tegangan keluarannya akan digunakan sebagai tegangan masukan

Arduino. Tegangan masukan Arduino memiliki range 9-12 Volt DC dan *Power Supply* ini dapat mengubah tegangan 220 VAC menjadi 12 VDC . Pengujian ini menggunakan multimeter "SANWA CD800a" untuk mengukur besar tegangan *output Power Supply*. Pengukuran ini untuk mendapatkan besar arus, tegangan dan bentuk sinyal yang keluar dari *Power Supply* pad Relai Utama 1 dan Relai Utama 2. Gambar 4.5 berikut ini adalah skema pengambilan data untuk *Power Supply*..

Gambar 4.5 **Skema Pengujian *Power Supply***

Setelah menyusun peralatan pengujian seperti skema diatas, dilakukan pengukuran arus dan tegangan pada *power supply* sebagai sumber arduino. Berikut adalah hasil pengujian *Power Supply* Relai Utama 1 dan Relai Utama 2 dapat dilihat pada tabel 4.2 dan 4.3

Tabel 4.2 Pengujian *Power Supply* Relai Utama 1

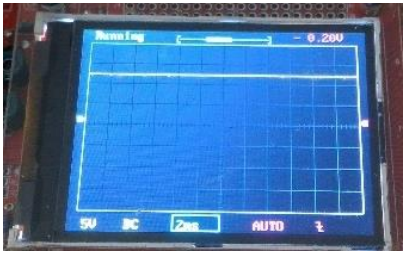
No	Parameter Pengujian	Hasil	Satuan
1	Tegangan	12,14	Volt
2	Arus	0,95	mA

Tabel 4.3 Pengujian *Power Supply* Relai Utama 2

No	Parameter Pengujian	Hasil	Satuan
1	Tegangan	12,16	Volt
2	Arus	0,95	mA

Untuk dapat membuktikan bahwa keluaran *power supply* merupakan gelombang DC, maka dibuktikan dengan melihat bentuk gelombang pada osiloskop. Gambar 4.6 dan gamabr 4.7 berikut adalah bentuk gelombang dari *power supply* yang ada pada Relai Utama 1 dan Relai Utama 2.

Gambar 4.6 **Bentuk Gelombang *Power Supply* Relai Utama 1**

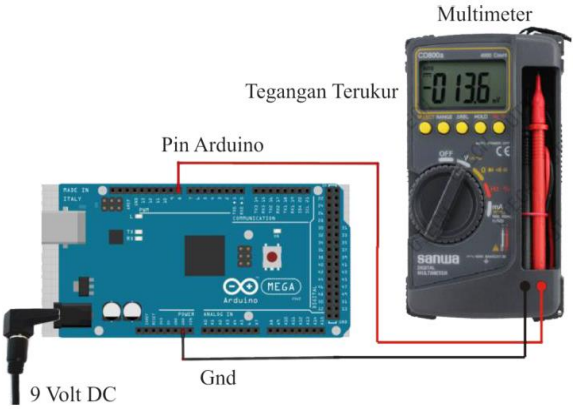


Gambar 4.7 **Bentuk Gelombang *Power Supply* Relai Utama 2**

Dari data yang telah diambil, dapat disimpulkan bahwa tegangan yang keluar dari *Power Supply* sebesar 12 Volt dengan error sebesar 14-16 mV. Arus yang diperoleh sama yaitu sebesar 0,95 mA. Dan bentuk gelombang dari *output Power Supply* berupa DC murni. Berikut adalah cara pengambilan data Power Supply yang ditunjukan oleh gambar 4.7

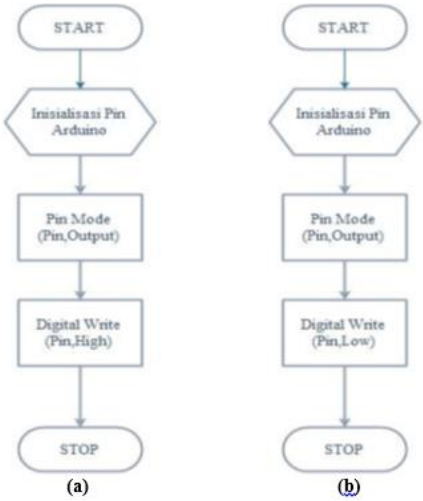
4.3 Pengujian *Input/Output Arduino*

Pegujian ini dilakukan terhadap *board* Arduino yang digunakan yakni Arduino Mega untuk relai utama dan Arduino Uno untuk relai indikator. Pengujian ini bertujuan untuk mengetahui bahwa kondisi Arduino Mega dapat digunakan dengan baik untuk Tugas Akhir ini. Skema pengujian ditunjukkan pada Gambar 4.9



Gambar 4.8 **Skema Pengujian Pin *Input/Output* Arduino**

Pengujian dilakukan dengan cara memberikan program pada Arduino yakni memberikan perintah *HIGH* dan *LOW* atau logika 1 dan 0 pada setiap pin Arduino yang akan diuji sesuai dengan *flowchart* yang ditunjukkan pada Gambar 4.9, kemudian mengukur besaran tegangan yang keluar dari pin tersebut seperti yang ditunjukkan pada Gambar 4.8. gambar 4.10 berikut adalah *flowchart* untuk pengujian tegangan pin Arduino Mega Dan Arduino Uno.



Gambar 4.9 ***Flowchart* Program Pengujian Pin Arduino (a) *Logic* 1, (b) *Logic 0***

Pada pengujian pin arduino ini dibagi menjadi 2 pengujian. Yaitu untuk relai utama 1 dan relai utama 2. Tabel 4.4 dan 4.5 berikut adalah hasil pengujian pin arduino :

Tabel 4.4 Pengujian Relai Utama 1

NO	NO PIN ARDUINO	*LOGIC*	TEGANGAN TERUKUR	*LOGIC*	TEGANGAN TERUKUR
1	A0	1	4.95 V	0	0.6 mv
2	A1	1	4.95 V	0	0.6 mv
3	A2	1	4.95 V	0	0.6 mv
4	A3	1	4.95 V	0	0.6 mv
5	A4	1	4.95 V	0	0.6 mv

NO	NO PIN ARDUINO	*LOGIC*	TEGANGAN TERUKUR	*LOGIC*	TEGANGAN TERUKUR
6	A5	1	4.95 V	0	0.7 mv
7	D0	1	4.95 V	0	96.9 mv
8	D1	1	4.95 V	0	4.7 mv
9	D2	1	4.95 V	0	1.2 mv
10	D3	1	4.95 V	0	1.2 mv
11	D4	1	4.95 V	0	1.3 mv
12	D5	1	4.95 V	0	1.4 mv
13	D6	1	4.95 V	0	0.7 mv
14	D7	1	4.95 V	0	0.7 mv
15	D8	1	4.95 V	0	0.8 mv
16	D9	1	4.95 V	0	0.8 mv
17	D10	1	4.95 V	0	0.9 mv
18	D11	1	4.95 V	0	0.9 mv
19	D12	1	4.95 V	0	0.8 mv
20	D13	1	4.95 V	0	0.8 mv
21	D18	1	4.95 V	0	1.7 mv
22	D19	1	4.95 V	0	1.6 mv
23	D20	1	4.95 V	0	11.8 mv
24	D21	1	4.95 V	0	11.6 mv
25	D35	1	4.95 V	0	0.9 mv
26	D37	1	4.95 V	0	1.1 mv

NO	NO PIN ARDUINO	*LOGIC*	TEGANGAN TERUKUR	*LOGIC*	TEGANGAN TERUKUR
27	D39	1	4.95 V	0	0.5 mv
28	D41	1	4.95 V	0	1.2 mv
29	D50	1	4.95 V	0	0.9 mv
30	D51	1	4.95 V	0	0.9 mv
31	D52	1	4.95 V	0	0.9 mv
32	D53	1	4.95 V	0	0.8 mv
Vreff		5V		4.95 V	
		VIN		11.39 V	
		3.3V		3.325 V	

Tabel 4.5 Pengujian Relai Utama 2

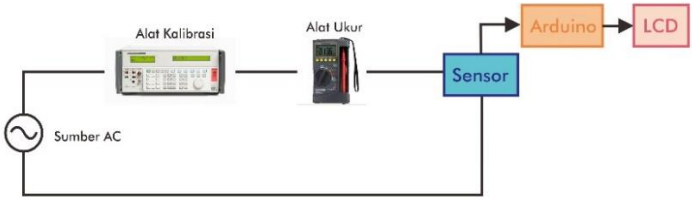
NO	NO PIN ARDUINO	*LOGIC*	TEGANGAN TERUKUR	*LOGIC*	TEGANGAN TERUKUR
1	A0	1	5.01 V	0	0.7 mv
2	A1	1	5.01 V	0	0.7 mv
3	A2	1	5.01 V	0	0.7 mv
4	A3	1	5.01 V	0	0.7 mv
5	A4	1	5.01 V	0	0.7 mv
6	A5	1	5.01 V	0	0.7 mv
7	D0	1	5.01 V	0	99.5 mv
8	D1	1	5.01 V	0	4.8 mv
9	D2	1	5.01 V	0	1.3 mv
10	D3	1	5.01 V	0	1.3 mv

NO	NO PIN ARDUINO	*LOGIC*	TEGANGAN TERUKUR	*LOGIC*	TEGANGAN TERUKUR
11	D4	1	5.01 V	0	1.4 mv
12	D5	1	5.01 V	0	1.5 mv
13	D6	1	5.01 V	0	0.7 mv
14	D7	1	5.01 V	0	0.8 mv
15	D8	1	5.01 V	0	0.9 mv
16	D9	1	5.01 V	0	0.8 mv
17	D10	1	5.01 V	0	0.9 mv
18	D11	1	5.01 V	0	0.9 mv
19	D12	1	5.01 V	0	0.9 mv
20	D13	1	5.01 V	0	0.9 mv
21	D18	1	5.01 V	0	1.7 mv
22	D19	1	5.01 V	0	1.7 mv
23	D20	1	5.01 V	0	12.3 mv
24	D21	1	5.01 V	0	12.3 mv
25	D35	1	5.01 V	0	1.1 mv
26	D37	1	5.01 V	0	1.2 mv
27	D39	1	5.01 V	0	0.6 mv
28	D41	1	5.01 V	0	1.3 mv
29	D50	1	5.01 V	0	0.9 mv
30	D51	1	5.01 V	0	0.9 mv
31	D52	1	5.01 V	0	0.9 mv
32	D53	1	5.01 V	0	0.9 mv

Vreff	5V	5.01 V
	VIN	11.41 V
	3.3V	3.32 V

4.4 Pembacaan Sensor Arus

Pada pengujian sensor arus ini kami melakukan kalibrasi sensor arus untuk mendapatkan tingkat akurasi pembacaan arus sebaik mungkin. Pengujian dilakukan kepada sensor arus SCT013-10 yang kami gunakan yang dengan menggunakan alat kalibrasi FLUKE 5500A. Berikut adalah gambar skema pengambilan data sensor arus.



Gambar 4.10 Skema Pengujian Sensor Arus

Pengujian sensor SCT013-10 dilakukan pada tiap-tiap relai, yaitu relai utama 1 dan relai utama 2. Proses pengambilan data di lakukan di ruang AJ302 , pada tanggal 29 mei 2017 ,dengan menggunakan FLUKE 5500A .Nilai kalibrasi didapatkan dari sebuah persamaan dari pengujian awal sensor arus pada tabel 4.6 dengan menggunakan rumus *scatter* pada *Microsoft Excel*. Pada persamaan 4.1 dapat diketahui besar nilai *error* (%) yang ada. Berikut ini adalah pengujian untuk mendapatkan nilai persamaan kalibrasi.

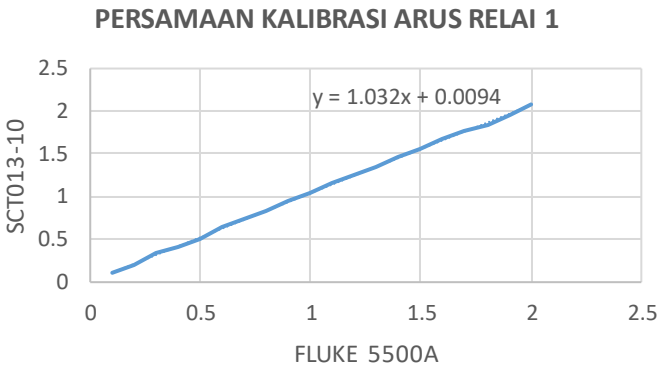
$$Error = \left|\frac{Hasil\ Sensor - Hasil\ Alat\ Ukur}{Hasil\ Alat\ Ukur}\right| x\ 100\% \quad \text{.....(4.1)}$$

Tabel 4.6 Pengujian untuk Persamaan Kalibrasi Relai 1

KALIBRASI ARUS RELAY 1			ERROR (%)
NO	FLUKE 5500A	SCT013-10	
1	0,1	0,11	10,00

KALIBRASI ARUS RELAY 1			ERROR (%)
NO	FLUKE 5500A	SCT013-10	
2	0,2	0,21	5,00
3	0,3	0,33	10,00
4	0,4	0,42	5,00
5	0,5	0,51	2,00
6	0,6	0,64	6,67
7	0,7	0,73	4,29
8	0,8	0,83	3,75
9	0,9	0,94	4,44
10	1	1,04	4,00
11	1,1	1,15	4,55
12	1,2	1,25	4,17
13	1,3	1,35	3,85
14	1,4	1,46	4,29
15	1,5	1,57	4,67
16	1,6	1,67	4,37
17	1,7	1,77	4,12
18	1,8	1,84	2,22
19	1,9	1,96	3,16
20	2	2,08	4,00
RATA-RATA ERROR (%)			**4,73**

Dari data diatas dapat diketahui grafik garis dan persamaannya. Gambar 4.12 berikut adalah hasil grafik garis beserta persamaannya :



Gambar 4.11 Grafik Garis dan Persamaan Kalibrasi Arus Relai 1

Untuk mendapatkan nilai kalibrasi, maka nilai awal kalibrasi pada program SCT013-10 di subtitusikan ke persamaan garis y = 1.032x + 0.0094.Karena nilai kalibrasi awal adalah 10, maka dapat dihitung sebagai berikut:

y = 1,032x + 0,0094.....(4.2)

10 = 1,032x + 0,0094 .....(4.3)

x = 9.6808139534883720930232558139535

Setelah didapatkan nilai x, pada program sensor arus nilai kalibrasi diubah sesuai dengan nilai x. Maka akan didapatkan pengukuran sensor arus baru seperti pada tabel 4.7 .

Tabel 4.7 Pengujian untuk Data Arus Relai 1

DATA ARUS RELAY 1			ERROR (%)
NO	FLUKE 5500A	SCT013-10	
1	0,1	0,10	0,00
2	0,2	0,20	0,00
3	0,3	0,30	0,00
4	0,4	0,40	0,00

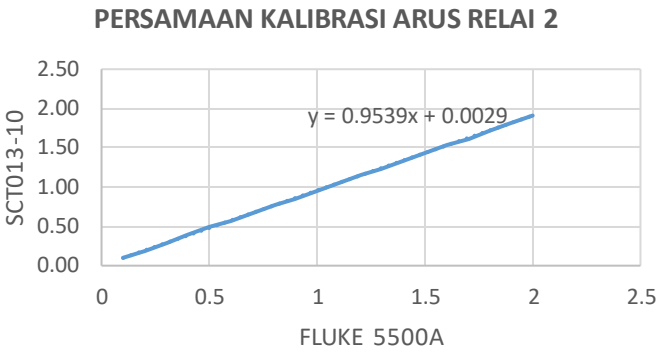
DATA ARUS RELAY 1			ERROR (%)
NO	FLUKE 5500A	SCT013-10	
5	0,5	0,51	2,00
6	0,6	0,60	0,00
7	0,7	0,71	1,43
8	0,8	0,79	1,25
9	0,9	0,91	1,11
10	1	1,01	1,00
11	1,1	1,11	0,91
12	1,2	1,21	0,83
13	1,3	1,29	0,77
14	1,4	1,41	0,71
15	1,5	1,51	0,67
16	1,6	1,60	0,00
17	1,7	1,72	1,18
18	1,8	1,80	0,00
19	1,9	1,90	0,00
20	2	2,01	0,50
RATA-RATA ERROR (%)			**0,62**

Dapat dilihat pada rata-rata *error*, sebelum dan sesudah mendapat nilai kalibrasi yang benar. Pada saat sebelum mendapat nilai kalibasi *error* sebesar 4,73%. Sedangkan sesudah mendapat nilai kalibrasi, *error* mengecil menjadi 0,42%. Untuk pengujian relai utama 2 juga dilakukan hal yang sama, dengan memberi nilai kalibrasi awal sebesar 10 dan nantinya akan diubah sesuai hasil yang didapatkan dari persamaan garis. Tabel 4.8 berikut adalah tabel persamaan kalibrasinya :

Tabel 4.8 Pengujian untuk Persamaan Kalibrasi Relai 2

KALIBRASI ARUS RELAY 2			
NO	**FLUKE 5500A**	**SCT013-10**	**ERROR (%)**
1	0,1	0,10	0,00
2	0,2	0,19	5,00
3	0,3	0,29	3,33
4	0,4	0,38	5,00
5	0,5	0,48	4,00
6	0,6	0,58	3,33
7	0,7	0,67	4,29
8	0,8	0,77	3,75
9	0,9	0,86	4,44
10	1	0,96	4,00
11	1,1	1,05	4,55
12	1,2	1,15	4,17
13	1,3	1,24	4,62
14	1,4	1,34	4,29
15	1,5	1,43	4,67
16	1,6	1,53	4,38
17	1,7	1,62	4,71
18	1,8	1,72	4,44
19	1,9	1,82	4,21
20	2	1,91	4,50
RATA-RATA ERROR			**4,08**

Dari data diatas dapat diketahui grafik garis dan persamaannya. Gambar 4.13 berikut adalah hasil grafik garis beserta persamaannya :



Gambar 4.12 Grafik Garis dan Persamaan Kalibrasi Arus Relai 2

Untuk mendapatkan nilai kalibrasi, maka nilai awal kalibrasi pada program SCT013-10 di subtitusikan ke persamaan garis y = 0.9539x + 0.0029. Karena nilai kalibrasi awal adalah 10, maka dapat dihitung sebagai berikut:

y = 0,9539x + 0,0029 ..... (4.4)

10 = 0,9539x + 0,0029 ..... (4.5)

x = 10.480239018765069713806478666527

Setelah didapatkan nilai x, pada program sensor arus nilai kalibrasi diubah sesuai dengan nilai x. Maka akan didapatkan pengukuran sensor arus baru seperti pada tabel 4.9.

Tabel 4.9 Pengujian untuk Data Arus Relai 2

DATA ARUS RELAY 2			ERROR (%)
NO	FLUKE 5500A	SCT013-10	
1	0,1	0,10	0,00
2	0,2	0,20	0,00
3	0,3	0,30	0,00
4	0,4	0,40	0,00
5	0,5	0,50	0,00
6	0,6	0,60	0,00

DATA ARUS RELAY 2			ERROR (%)
NO	FLUKE 5500A	SCT013-10	
7	0,7	0,70	0,00
8	0,8	0,80	0,00
9	0,9	0,90	0,00
10	1	1,00	0,00
11	1,1	1,10	0,00
12	1,2	1,20	0,00
13	1,3	1,30	0,00
14	1,4	1,40	0,00
15	1,5	1,50	0,00
16	1,6	1,60	0,00
17	1,7	1,70	0,00
18	1,8	1,80	0,00
19	1,9	1,90	0,00
20	2	2,00	0,00
RATA-RATA ERROR			**0,00**

Dapat dilihat pada rata-rata *error*, sebelum dan sesudah mendapat nilai kalibrasi yang benar. Pada saat sebelum mendapat nilai kalibasi *error* sebesar 4,08%. Sedangkan sesudah mendapat nilai kalibrasi, *error* mengecil menjadi 0,00% atau tidak terdapat *error*. Dengan begitu, sensor arus ini sudah terkalibrasi dan lebih akurat dari sebelumnya, karena tidak memiliki nilai *error* yang lebih kecil. Gambar 4.14 berikut adalah proses pengujian sensor arus

Gambar 4.13 Proses Pengujian Sensor Arus

4.5 Pembacaan Sensor Tegangan

Pada pengujian sensor tegangan ini kami melakukan kalibrasi sensor tegangan untuk mendapatkan tingkat akurasi pembacaan arus sebaik mungkin. Proses pengambilan data di lakukan di ruang AJ302 , pada tanggal 29 mei 2017 ,dengan menggunakan FLUKE 5500A Pengujian dilakukan kepada trafo CT 350 mA yang kami gunakan yang dengan menggunakan alat kalibrasi FLUKE 5500A. Gambar 4.14 berikut adalah gambar skema pengambilan data sensor arus.

Gambar 4.14 Skema Pengujian Sensor Tegangan

Pengujian sensor tegangan ini dilakukan pada tiap-tiap relai, yaitu relai utama 1 dan relai utama 2. Dengan persamaan 4.1 dapat diketahui besar nilai *error* (%) yang ada. Tabel 4.10 dan 4.11 berikut ini adalah pengujian untuk sensor tegangan relai 1 dan relai 2.

Tabel 4.10 Pengujian untuk Data Tegangan Relai 1

DATA SENSOR TEGANGAN RELAY 1			
NO	**FLUKE 5500A**	**SCT013-10**	**ERROR(%)**
1	100	100,62	0,62
2	110	110,72	0,65
3	120	120,74	0,62
4	130	130,77	0,59
5	140	140,98	0,70
6	150	150,40	0,27
7	160	161,19	0,74
8	170	170,99	0,58
9	180	181,12	0,62
10	190	191,04	0,55
11	200	201,25	0,63
12	210	211,25	0,60
13	220	221,28	0,58
14	230	231,14	0,50
15	240	240,43	0,18
16	250	250,78	0,31
RATA-RATA ERROR			**0,55**

Tabel 4.11 Pengujian untuk Data Tegangan Relai 2

DATA SENSOR TEGANGAN RELAY 2			
NO	**FLUKE 5500A**	**SCT013-10**	**ERROR (%)**
1	100	100,24	0,24
2	110	110,12	0,11
3	120	120,30	0,25

DATA SENSOR TEGANGAN RELAY 1			
NO	**FLUKE 5500A**	**SCT013-10**	**ERROR(%)**
4	130	130,52	0,40
5	140	140,39	0,28
6	150	150,63	0,42
7	160	160,84	0,53
8	170	170,86	0,51
9	180	180,97	0,54
10	190	190,86	0,45
11	200	201,31	0,66
12	210	211,21	0,58
13	220	221,70	0,77
14	230	231,42	0,62
15	240	241,63	0,68
16	250	251,03	0,41
RATA-RATA ERROR			**0,46**

Dari data yang telah diperoleh dari pengujian sensor tegangan, dapat diketahui bahwa nilai *error* sensor tegangan pada relai 1 sebesar 0,55% dan pada relai 2 sebesar 0,46%. Dengan begitu, sensor tegangan ini telah terkalibrasi, karena tidak memiliki nilai *error* yang besar dengan eror di bawah 1%.

4.6 Pengujian Arus Impedansi Tinggi

Untuk melakukan uji short circuit jaringan akan mengalami perubahan arus yang sangat besar , apabila pada suatu bangunan terdapat mcb maka mcb akan trip secara otomatis terlebih dahulu Karena setting waktu mcb sangat cepat dan arus nominalnya tergolong kecil sehigga rele tidak dapat di amati dengan baik. Oleh Karena itu di buat suatu rangkaian impedansi guna membatasi arus short circuit agar kerja dari rele dapat di amati dengan jelas .untuk mencapai arus short yang rendah di butuh kan

impedansi sebesar 100 ohm seperti gambar 4.16,berikut merupakan perhitungan dari impedansi short circuit:

V = I x R
220 = I x 100
I = 2,2 Ampere

Gambar 4.15 **Pengujian Impedansi Short Circuit**

Keterangan :
V = tegangan ac pada jaringan (dengan asumsi tegangan 220V)
I = nilai arus AC yang mengalir pada rangkaian
R = nilai impedansi pada rangkaian

Berdasarkan perhitungan jika menggunakan impedansi sebesar 100 ohm dengan tegangan 220 maka arus short circuit sebesar 2,2 Ampere sehingga ketika melakukan pengujian dengan mcb 5 ampere rele masih dapat di amati dengan jelas.

4.7 Pengujian Karakteristik Over Current Dan Short Circuit Relay

Untuk mengetahui apakah setting yang telah di atur oleh Lazarus telah sesuai pada setting digital relay maka di perlukan pengujian karakteristik waktu rele. Proses dalam pengambilan data yaitu relay akan di setting dengan nilai tertentu , relay akan di beri beberapa pembebanan dan kombinasinya ,data berupa nilai arus dan waktu trip akan di olah dalam excel sehingga nilai dari waktu trip perhitungan dan waktu trip pengukuraan dapat di bandingkan dalam bentuk grafik .

Dalam pengambilan data akan digunakan dua jenis kurva yaitu normal inverse dan devinite, Untuk melakukan pengecekan pada kedua relay maka pengaturan Isett , Isc, Waktu trip dan Waktu TMS di buat sama dengan nilai masing masing;

Iset : 1.5 A
Isc : 2.3 A
Waktu trip: 0 s
Waktu Tms :0.1 s

Sedangkan untuk pengetesan di perlukan pembebanan , terdapat beberapa jenis dan kombinasi pembebanan agar arus yang terukur dapat di amati, , adapun beban beban yang di gunakan sebagai berikut

1. Setrika (350 watt)
2. Lampu (25 watt)
3. Lampu (100 watt)
4. Trip modul (440 watt)

Table 4.12 berikut merupakan hasil dari data yang di dapat pada relay 1 dan 2 beserta perbandingan dengan perhitungan waktu trip.

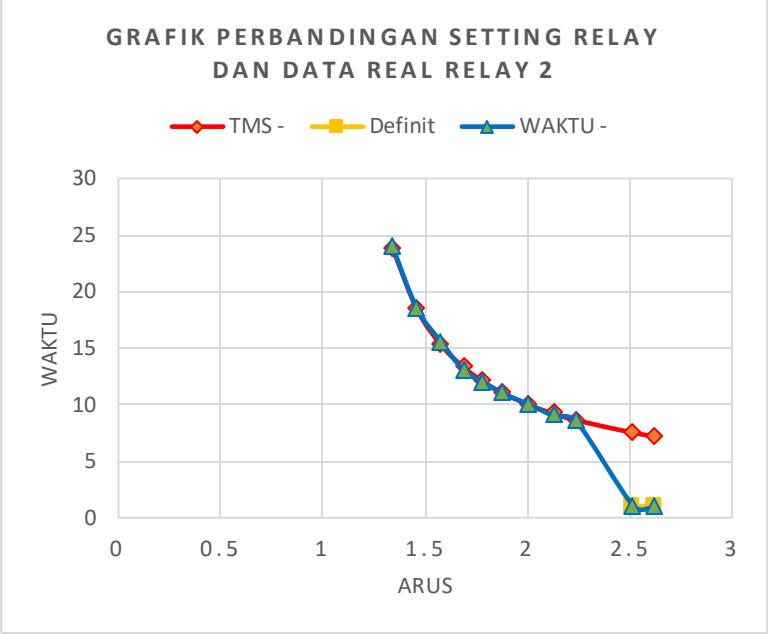
Tabel 4.12 **Hasil Uji Relay 2**

SET RELAI 2 [OL = (Iset=1A ; Timedial=1s), SC =(Isc=2.3A ; Timeset= 0.5s)]							
NO	Beban		**I**	Setting		Data Nyata	
	Beban 1	**Beban 2**	**NOM**	**TMS**	**Definite**	**WAKTU**	**KONDISI**
1	lampu 200 watt		**0.85**	-		-	normal
2	rice cooker (cook) 300 watt		1.34	23.84782287		24	trip
3	rice cooker (cook) 300 watt	lampu 25 watt	1.46	18.42725015		18.5	trip
4	rice cooker (cook) 300 watt	lampu 50 watt	1.58	15.233167		15.5	trip

SET RELAI 2 [OL = (Iset=1A ; Timedial=1s), SC =(Isc=2.3A ; Timeset= 0.5s)]							
NO	BEBAN		I NOM	Setting		Data Nyata	
	Beban 1	**Beban 2**		**TMS**	**Definite**	**WAKTU**	**KONDISI**
5	rice cooker (cook) 300 watt	lampu 75 watt	1.69	13.27035384		13	trip
6	rice cooker (cook) 300 watt	lampu 100 watt	1.78	12.06998498		12	trip
7	rice cooker (cook) 300 watt	lampu 125 watt	1.88	11.01887367		11	trip
8	rice cooker (cook) 300 watt	lampu 150 watt	2.01	9.95688092		10	trip
9	rice cooker (cook) 300 watt	lampu 175 watt	2.13	9.187941957		9	trip
10	rice cooker (cook) 300 watt	lampu 200 watt	2.24	8.609927145		8.5	trip
11	impedansi 440 watt	lampu 25 watt	2.52	7.503851129	1	1	trip
12	impedansi 440 watt	lampu 50 watt	2.62	7.19786038	1	1	trip

Berdasarkan tabel di atas beberapa beban yang di kombinasikan akan membuat urutan arus yang semakin tinggi , yang memiliki beberapa karakter dimana arus yang bernilai si bawah 1 ampere tidak akan membuat trip relay 2, sedangkan arus yang bernilai di atas 1 ampere akan membuat relay 2 dalam kondisi trip. waktu trip yang di peroleh dari nilai arus diatas 1 ampere dan di bawah 2.3 ampere memiliki waktu yang berbeda beda. Semakin tinggi nilai arus pada nilai 1 sampai 2.3 ampere maka waktu trip semakin cepat , sedangkan waktu trip di atas 2.3 memiliki waktu yang sama , untuk membuktikan apakah nilai dari data arus dan waktu trip sudah sesuai dengan kurva setting arus inverse dan devinite

maka di perlukan pembandingan data dengan menggunakan grafik, grafik merupakan hasil dari perbadingan data Relay 2



Gambar 4.16 **Grafik Perbandingan Setting Relay Dan Data Real Relay 2**

Berdasarkan **gambar 4.17** grafik di atas nilai waktu trip relay 2 antara 1 Ampere sampai dengan 2.3 ampere tidak terlalu jauh dari nilai setting, sehingga data yang di dapat sesuai dengan karakteristik dari kurva invers, sedangkan waktu trip relay gangguan yang lebih dari 2.3 ampere memiliki waktu kerja yang sama dengan karakteristik kurva definite

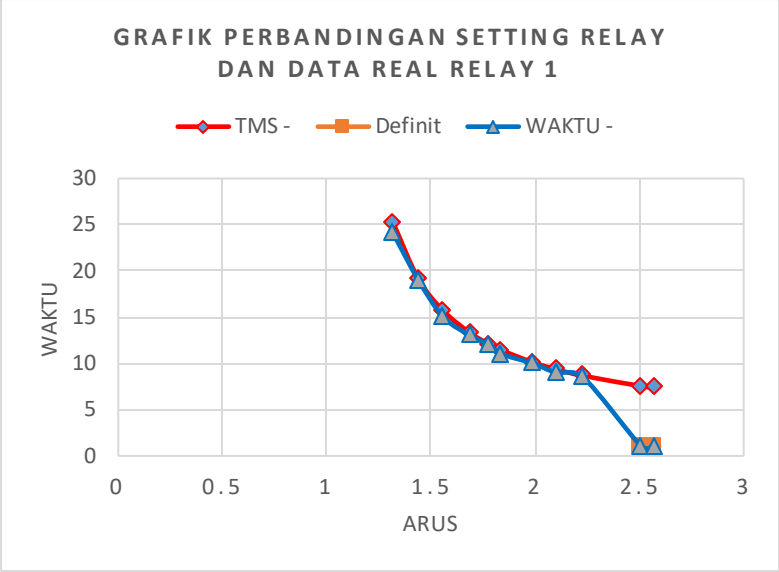
Setelah melakukan pengambilan data pada relay 2 maka langkah selanjutnya adalah melakukan pengambilan data pada relay 1 dengan nilai setting yang sama,tabel 4.13 berikut merupakan hasil dari pengambilan data relay 1

Tabel 4.13 **Hasil Uji Relay 1**

SET RELAI 2 [OL = (Iset=1A ; Timedial=1s), SC =(Isc=2.3A ; Timeset= 0.5s)]							
NO	Beban		**I**	Setting		Data Nyata	
	Beban 1	**Beban 2**	**NOM**	**TMS**	**Definite**	**WAKTU**	**KONDISI**
1	lampu 200 watt		**0.88**	-		-	normal
2	rice cooker (cook) 300 watt		1.32	25.14332061		24	Trip
3	rice cooker (cook) 300 watt	lampu 25 watt	1.44	19.1269374		19	Trip
4	rice cooker (cook) 300 watt	lampu 50 watt	1.56	15.67155461		15	Trip
5	rice cooker (cook) 300 watt	lampu 75 watt	1.69	13.27035384		13	Trip
6	rice cooker (cook) 300 watt	lampu 100 watt	1.78	12.06998498		12	trip
7	rice cooker (cook) 300 watt	lampu 125 watt	1.84	11.40996391		11	trip
8	rice cooker (cook) 300 watt	lampu 150 watt	1.99	10.10258847		10	trip
9	rice cooker (cook) 300 watt	lampu 175 watt	2.1	9.364932063		9	trip
10	rice cooker (cook) 300 watt	lampu 200 watt	2.23	8.658349429		8.5	trip

SET RELAI 2 [OL = (Iset=1A ; Timedial=1s), SC =(Isc=2.3A ; Timeset= 0.5s)]							
NO	Beban		I NOM	Setting		Data Nyata	
	Beban 1	Beban 2		TMS	Definite	WAKTU	KONDISI
11	impedansi 440 watt	lampu 25 watt	2.51	7.536572643	1	1	Trip
12	impedansi 440 watt	lampu 50 watt	2.57	7.346213732	1	1	Trip

Berdasarkan tabel 4.13 di atas beberapa beban yang di kombinasikan akan membuat urutan arus yang semakin tinggi , yang memiliki beberapa karakter dimana arus yang bernilai si bawah 1 ampere tidak akan membuat trip relay 1, sedangkan arus yang bernilai di atas 1 ampere akan membuat relay 1 dalam kondisi trip. waktu trip yang di peroleh dari nilai arus diatas 1 ampere dan di bawah 2.3 ampere memiliki waktu yang berbeda beda. Semakin tinggi nilai arus pada nilai 1 sampai 2.3 ampere maka waktu trip semakin cepat , sedangkan waktu trip di atas 2.3 memiliki waktu yang sama , untuk membuktikan apakah nilai dari data arus dan waktu trip sudah sesuai dengan kurva setting arus inverse dan devinite maka di perlukan pembandingan data dengan menggunakan grafik, grafik merupakan hasil dari perbadingan data Relay 1.



Gambar 4.17 **Grafik Perbandingan Setting Relay Dan Data Real Relay 1**

Berdasarkan **Gambar 4.18** grafik di atas nilai waktu trip relay 1 antara 1 Ampere sampai dengan 2.3 ampere tidak terlalu jauh dari nilai setting, sehingga data yang di dapat sesuai dengan karakteristik dari kurva invers, sedangkan waktu trip relay gangguan yang lebih dari 2.3 ampere memiliki waktu kerja yang sama dengan karakteristik kurva definite.

4.8 Pengujian Koordinasi *Relay*

Dalam sistem proteksi ,digital relay digunakan sebagai intrumen pengatur jalannya proteksi oleh karena itu koordinasi tiap relay sangat di perlukan agar sistem distribusi tenaga listrik lebih efisien saat terjadi gangguan Dalam proses proteksi terdapat dua jenis gangguan yaitu over current dan short circuit, pada prototipe sistem proteksi ini terdapat 2 buah relay di mana relay-relay tersebut saling terkoordinasi. Dalam proses koordinasi relay pengaturan nilai arus dan waktu sangatlah penting , oleh karena itu di butuhkan beberapa data koordinasi antar relay yaitu overcurrent tes dan short circuit tes

Over current tes di lakukan dengan 2 cara yaitu yang pertama kondisi dimana time dial kedua relay sama namun setting arus over

current berbeda , sedangkan yang kedua lakukan dengan cara mensettingan time dial kedua relay berbeda namun arus over current sama. Hasil dari tes over curren di tampilkan dalam bentuk **tabel** 4.14 seperti berikut

Tabel 4.14 **Data pengujian koordinasi over current test**

over current test														
no	relay 1		relay 2		beban		I nom		waktu trip		waktu hitung		kondisi	
	Iset	timedial	Iset	timedial	busbar 1	busbar 2	relay 1	relay 2	relay 1	relay 2	relay 1	relay 2	relay 1	relay 2
1	1	1.2	1	0.6	-	1.35	1.33	1.35	30	14	29.4	13.95	-	TRIP
2	1	1.2	1	0.6	1.34	-	1.34	-	29	-	28.6	-	TRIP	-
3	1	1.2	1	0.6	0.74	1.36	2.07	1.36	10	12	11.5	13.62	TRIP	-
4	1	1.2	1	0.6	1.32	0.75	2.07	0.75	10	-	11.5	-	TRIP	-
5	1.2	0.6	1	0.6	-	1.36	1.34	1.36	38	13.5	38	13.62		TRIP
6	1.2	0.6	1	0.6	1.35	-	1.35	-	36	-	35.6	-	TRIP	-
7	1.2	0.6	1	0.6	0.72	1.35	2.06	1.35	8	14	7.73	13.95	TRIP	-
8	1.2	0.6	1	0.6	1.31	0.75	2.06	0.75	8	-	7.73	0	TRIP	-

Tes pengujian berikutnya yaitu short circuit tes. Short circuit tes di lakukan dengan 2 cara yaitu yang pertama kondisi dimana timetrip kedua relay sama namun setting arus short circuit berbeda , sedangkan yang kedua di lakukan dengan cara mensettingan timetrip kedua relay berbeda namun arus short circuit sama. Hasil dari Short circuit tes di tampilkan dalam bentuk **tabel 4.15** seperti berikut :

Tabel 4.15 **Data pengujian koordinasi short circuit tes**

short circuit test												
no	relay 1		relay 2		beban		I nom		waktu trip		kondisi	
	Iset	time set	I set	time set	busbar 1	busbar 2	busbar 1	busbar 2	relay 1	relay 2	relay 1	relay 2
1	2.1	2	2.1	1	-	short (2.21A)	2.21	2.21		1		trip
2	2.1	2	2.1	1	short (2.22A)	0.77	2.97	0.77	2		trip	
3	2.5	1	2.3	1	0.26	short (2.21A)	2.48	2.22		1		trip
4	2.5	1	2.3	1	short (2.22A)	0.39	2.57	0.39	1		trip	

Keterangan:

Beban busbar 1 = beban yang ada pada busbar 1 (Ampere)

Beban busbar 2 = beban yang ada pada busbar 2 (Ampere)

I nom relay = arus yang mengalir pada relay 1(Ampere)

I nom relay = arus yang mengalir pada relay 1(Ampere)

Waktu trip relay 1= pembacaan waktu yang di butuhkan relay 1 untuk trip dengan mengunakan stopwatch (detik)

Waktu trip relay 2 = pembacaan waktu yang di butuhkan relay 2 untuk trip dengan mengunakan stopwatch (detik)

Waktu hitung relay 1 = perhitungan waktu yang di butuhkan relay 1 untuk trip berdasarkan perhitungan kurva proteksi inverse (detik)

Waktu hitung relay 2 = perhitungan waktu yang di butuhkan relay 2 untuk trip berdasarkan perhitungan kurva proteksi inverse (detik)

Kondisi relay 1 = kondisi relay 1 setelah ada gangguan pada jaringan

Kondisi relay 2 = kondisi relay 2 setelah ada gangguan pada jaringan

4.9 Analisa Relevansi

Prototipe ini merupakan permodelan dari sistem proteksi yang ada pada sistem distribusi tenaga listrik PLN ,fungsi lebih dari alat ini adalah adanya sistem pengaturan yang dapat di lakukan dai jarak yang jauh dan di tempat yang sama..namun yang harus di sadari adalah keterbatasan nilai yang dapat di tampung sehingga tidak dapat di implementasikan langsung pada dunia kerja, namun apabila di implementasikan dalam sebuah sistem yang besar maka akan memepermudah kinerja para pagawai dalam melakukan pengaturan ulang sistem proteksi. Oleh Karena itu di perlukan penelitian lebih guna menyempurnakan alat ini

BAB V
PENUTUP

Bab penutup ini berisi kesimpulan yang diperoleh selama proses pembuatan tugas akhir yang berjudul **"**DESAIN PROTOTIPE DAN KOORDINASI DIGITAL RELAY SATU FASA TERHADAP GANGGUAN OVERCURRENT**"**,serta saran untuk pengembangan alat ini kedepannya.

5.1 Kesimpulan

Setelah melakukan proeses pembuatan dan pengujian alat tugas akhir terdapat beberapa bahasan penting yang dapat di Tarik kesimpulan, adapun salah satunya yaitu

1. Proses pembacaan arus pada awal uji coba memiliki eror yang tinggi sekitar 4.73% namun setelah di lakukan kalibrasi nilai error berkurang menjadi 0.62%
2. Waktu yang di perlukan relay untuk trip pada kondisi over current memiliki nilai yang berbeda beda berdasarkan besarnya arus overload dan setting timedialnya
3. Waktu trip pada kondisi short ciruit memiliki waktu yang sama apabila kondisi arus yang di uji melewati nilai setting arus shortcircuit

5.2 Saran

Dalam suatu proses pembelajaran terdapat beberapa hal yang dapat menjadi saran bagi pembaca.Di harapkan nanti dapat di lamjutkan maupun di perbaiki menjadi lebih baik. Beberapa saran yang kami sarankan adalah:

1. Proses pengujian koordinasi relai ini nantinya dapat lebih disesuaikan dengan keadaan nyata dengan ditambahkan beberapa beban lebih besar dll.
2. Di harapkan menggunakan genset sebagai sumber tenaga, hal tersebut di harapkan mendekati kondisi nyata dari pembangkit.Selain itu dengan menggunakan genset dapat memperbesar batasan arus untuk penelitian yang lebih presisi.
3. Dan juga diharapkan alat ini nantinya dapat beroperasi pada jaringan 3 fasa agar dapat menyerupai alat yang ada di lapangan.

-----Halaman ini sengaja dikosongkan-----

DAFTAR PUSTAKA

[1] ...., **PENGESAHAN RUPTL PLN TAHUN 2016-2025**, MENTERI ENERGI DAN SUMBER DAYA MINERAL REPUBLIK INDONESIA, 2016

[2] Andrianto, Heri; dan Aan Darmawan. 2016. **Arduino Belajar Cepat dan Pemrograman**. Bandung : INFORMATIKA.

[3] ...., **IDE *Software for* Arduino**, Datasheet, 2015.

[4] ...., **INFORCE UNINTERRUPTIBLE POWER SUPPLY 650 VA**, Manual Book, 2009.

[5] DFROBOT, **ARDUINO LCD KEYPAD DFROBOT**, https://www.dfrobot.com/, 19 Mei 2017.

[6] …., **OPEN ENERGY MONITOR,** https://openenergymonitor.org/, 15 April 2017

[7] **AZZETTLER *RELAYS***, http://www.azettler.com, 16 April 2017

[8] Adhitya, W.W. dan Faisal, A., **PERANCANGAN SISTEM MONITORING VOLTAGE FLICKER BERBASIS ARDUINO DENGAN METODE FAST FOURIER TRANSFORM (FFT)**, Tugas Akhir, Program D3 Teknik Elektro FTI-ITS, Surabaya, 2016.

-----Halaman ini sengaja dikosongkan-----

LAMPIRAN A

A.1. Listing Program pada Arduino

```
/*ReleComb.h*/
// Pin Assigment
#define PIN_TRIP_SWITCH 39 // Test Switch
#define PIN_RESET_SWITCH 41 // Reset Switch
#define PIN_RELAY_OUT 11 // Relay Output
#define PIN_LED_LIVE 13 // LED live
#define PIN_TX_RX_ENABLE 3 // RS-485 enable pin
#define SLAVE_ADDR_PIN0 35 // Slave ID Address0
#define SLAVE_ID 2 // Address Modbus Slave
#define ALAMAT_EEPROM_AWAL_EVENT 101 // Address Awal Untuk Data Logger Gangguan
#define ALAMAT_EEPROM_AWAL_DATA 0 // Address Awal Untuk Data Relay
#define JUMLAH_EVENT_MAX 50 // Jumlah Maksimal Data Logger Gangguan
#define CROSSING_MAX 2 // EmonLib Crossing 20 kali
#define TIME_OUT_MAX 50 // EmonLib Time Out 2000 ms
#define SHORT_CIRCUIT 1 //Status Short Circuit
#define OVER_LOAD 2 //Status Over Load
// Operational from Host
#define NO_OPERATION 0
#define DEFAULT_SET 1
#define CHANGE_SET 2
#define READ_SET 3
#define CLEAR_EVENT 4
#define READ_EVENT 5
#define TRIP_REMOTE 6
#define RESET_REMOTE 7
#define READ_HISTORY_SET 8

// LCD pin setting
// For LCD Keypad DF Robot
#define LCD_RS 8
#define LCD_EN 9
#define LCD_D4 4
#define LCD_D5 5
```

```
#define LCD_D6  6
#define LCD_D7  7
#define PIN_KEYPAD  A0 // pin Analog Keypad
#define LCD_ROWS  2
#define LCD_COLUMNS  16
//
#define BUTTONS_PER_COLUMN  5 // Each analog pin has five
buttons with resistors.
#define BUTTONS_PER_ROW  1 // There are two analog pins in use.

#define  REMOTE_RESET_ACTIVE  1//Reset dari Host
#define  REMOTE_TRIP_ACTIVE  1//Trip dari Host

#define  MAX_DETIK  60
#define  BAUD_RATE  9600 //untuk serial comm
#define  A20MILLISECOND  20 //20 ms
#define  SEQ_CYCLE_SPEED  20 //20 ms
#define  WORD_MAX  0xFFFF//maksimum 16 bit
#define  BYTE_MAX  0xFF//maksimum 8 bit
#define  MAX_PV  10.0 // Nilai Arus Maksimum
#define  MAX_TEG  500.0 // Nilai Teg Maksimum

// For Menu Index
#define MAIN_MENU_IDX  0 //Index Main Menu
//-----------------
#define TRIP_RESET_MENU_IDX  10 //Index Short Cicuit Menu
#define TRIP_MENU_IDX  1 //Index KeyPad Trip
#define RESET_MENU_IDX  2 //Index Short Cicuit Reset
#define BACK_MENU_IDX  3 //Index Short Cicuit Reset
//-----------------
#define DISPLAY_MENU_IDX  20 //Index Display Menu
//-----------------
#define STATUS_MENU_IDX  30 //Index Display Menu
//-----------------
#define INFO_MENU_IDX  40 //Index Display Menu

struct IdMenu
{
  byte mainMenu;
```

```
  byte subMenu;
};

struct DataEvent {
   byte   NomorEvent;
   float  Arus;
   float  Tegangan;
   byte   Status;
   byte   EventType;
   byte   Detik;
   byte   Menit;
   byte   Jam;
   byte   Tanggal;
   byte   Bulan;
   int    Tahun;
};

struct DataSettingRelay {
   byte   TChar;
   byte   Action;
   float  SetPoint;
   unsigned int  TMS;
   unsigned int  TSet;

   byte   Version;
   byte   Detik;
   byte   Menit;
   byte   Jam;
   byte   Tanggal;
   byte   Bulan;
   int    Tahun;
};
DataSettingRelay DataReleShortCircuit,DataReleOverLoad;

enum
{
  // just add or remove registers and your good to go...
  // The first register starts at address 0
//Data from Ardu
```

```
  ARDU_ID, //index 0
//For Short Circuit
  ARDU_SH_SP, //index 1
  ARDU_SH_PV, //index 2
  ARDU_SH_ACTION, //index 3
  ARDU_SH_TCHAR, //index 4
  ARDU_SH_TSET, //index 5
  ARDU_SH_TMS, //index 6
  ARDU_SH_TACC, //index 7
  ARDU_SH_STATE, //index 8
//For Over Load
  ARDU_OL_SP, //index 9
  ARDU_OL_PV, //index 10
  ARDU_OL_ACTION, //index 11
  ARDU_OL_TCHAR, //index 12
  ARDU_OL_TSET, //index 13
  ARDU_OL_TMS, //index 14
  ARDU_OL_TACC, //index 15
  ARDU_OL_STATE, //index 16
  ARDU_EVENT_NBR, //index 17
//Operational from Host
  HOST_OPR_MODE, //index 18
  HOST_OPR_STATUS, //index 19
  LAST_OPR_MODE, //index 20
//for setting data Short Circuit
  HOST_SH_SP, //index 21
  HOST_SH_ACTION, //index 22
  HOST_SH_TCHAR, //index 23
  HOST_SH_TSET, //index 24
  HOST_SH_TMS, //index 25
//for setting data Over Load
  HOST_OL_SP, //index 26
  HOST_OL_ACTION, //index 27
  HOST_OL_TCHAR, //index 28
  HOST_OL_TSET, //index 29
  HOST_OL_TMS, //index 30
//tgl 13/05/2017 --- for setting date
  HOST_SET_VERSION, //index 31
  HOST_SET_TAHUN, //index 32
```

```
  HOST_SET_BULAN,  //index 33
  HOST_SET_TGL,  //index 34
  HOST_SET_JAM,  //index 35
  HOST_SET_MENIT,  //index 36
  HOST_SET_DETIK,  //index 37
//tgl 13/05/2017  --- for event data
  HOST_EVENT_WILL_READ,  //index 38, nomor event yang akan
dibaca
  HOST_EVENT_NUMBER,  //index 39
  HOST_EVENT_TAHUN,  //index 40
  HOST_EVENT_BULAN,  //index 41
  HOST_EVENT_TGL,  //index 42
  HOST_EVENT_JAM,  //index 43
  HOST_EVENT_MENIT,  //index 44
  HOST_EVENT_DETIK,  //index 45
  HOST_EVENT_STATUS,  //index 46
  HOST_EVENT_ARUS,  //index 47
  HOST_EVENT_TEG,  //index 48
  HOST_EVENT_TYPE,  //index 49
  HOST_DIAG1,  //index 50
  HOST_DIAG2,  //index 51
// leave this one
  HOLDING_REGS_SIZE
  // total number of registers for function 3 and 16 share the same register
array
};
unsigned int holdingRegs[HOLDING_REGS_SIZE];  // function 3 and 16
register array

/*RelayArdu*/
/*
 *   Program ini dirancang untuk Relay  Proteksi
 *   - Version 0.0,  8/5/2017
*/
#if ARDUINO  < 100
#include <WProgram.h>
#else
#include <Arduino.h>
#endif
```

```
#include "EmonLib.h" //Library sensor arus dan tegangan
#include "ReleComb.h" //Untuk inisialisai pin dll
#include <RTClib.h> //Library RTC
#include <Wire.h> //Untuk 12C
#include <RelayProt.h> //Library relay proteksi
#include <SequenceTimer.h>//Library waktu millis

#include <SD.h> //Library SDCard
#include <EEPROM.h> //Library EEPROM
#include <LiquidCrystal.h> // Library LCD
#include <phi_interfaces.h> //Library keypad
#include <phi_prompt.h> //Library LCD display
#include <SimpleModbusSlave.h> //Library protokol Modbus Slave

volatile int itrNmbr; //niai interrupt
volatile boolean sudahTulisEvent; //pernyataan sudah menulis ke EEPROM

EnergyMonitor monitoringArusDanTegangan; //Create an instance for Emonlib

DataEvent DataTulis; //Buat menulis ke EEPROM
RTC_DS1307 rtc; //Deklarasi pake RTC DS1307
//RTC_Millis rtc; //Deklarasi buat ngambil waktu dari laptop, simulasi
DateTime SaatIni; //Waktu saat ini

unsigned long prevMilli; //Millis sebelumnya
byte keypad_type=Analog_keypad; //Memilih tipe keypad analog
char mapping[]={'R','U','D','L','S'}; //Right, Up, Down, Left, Select
byte pins[]={PIN_KEYPAD}; //The pin numbers are analog pin numbers
int values[]={5, 105, 260, 415, 640}; //value for DF Robot LCD Keypad
phi_analog_keypads analogKeypad(mapping, pins, values, BUTTONS_PER_ROW, BUTTONS_PER_COLUMN); //Mapping analogkeypad
multiple_button_input * keypads[]={&analogKeypad,0};
char up_keys[]={"U"}; ///< All keys that act as the Up key are listed here.
char down_keys[]={"D"}; ///< All keys that act as the Down key are listed here.
```

```
char left_keys[]={"L"}; ///< All keys that act as the Left key are listed here.
char right_keys[]={"R"}; ///< All keys that act as the Right key are listed here.
char enter_keys[]={"S"}; ///< All keys that act as the Select key are listed here.
char *
function_keys[]={up_keys,down_keys,left_keys,right_keys,enter_keys}
; ///< All function key names are gathered here fhr phi_prompt.
// LCD
LiquidCrystal
lcd(LCD_RS,LCD_EN,LCD_D4,LCD_D5,LCD_D6,LCD_D7);
//deklarasi pin LCD
int global_style=109; // This is the style of the menu
// SD Card
File myFile;
int pinCS = 53;
// Relay Proteksi
RelayProt RelayShortCirt("RlShortCirt"); //Deklarasi relay Short Circuit
RelayProt RelayOverLoad("RlOverLoad"); //Deklarasi relay Over Load
SequenceTimer SequenceUtama(SEQ_CYCLE_SPEED); //dalam mili detik

IdMenu menuId;

void setup() {
  // put your setup code here, to run once:
  // Timer0 is already used for millis() - we'll just interrupt somewhere
  // in the middle and call the "Compare A" function below
  OCR0A = 0xAF;
  TIMSK0 |= _BV(OCIE0A);
  prevMilli = millis();

  setupModbus (); //Setup Modbus, Addressing dan konfigurasi

  arduInterfaceSetup(); //Setup Keypad-LCD
  pinMode(PIN_LED_LIVE, OUTPUT); //LED pin 13

  SetupEmonlib(); //Setup baca arus dan tegangan
```

```
//Setup Relay
  setupRelayShortCirt(); //setup relay protection untuk short circuit
  setupRelayOverLoad(); //setup relay protection untuk overLoad, 25/05/2017
  setupDefault(); //Setting Default

  updateDataRelayFromEEPROM(); //Ambil Data Dari EEPROM Dan Validasi, saat pertama kali jalan
//.....
  readEventFromEEPROM(); //Ambil Event Dari EEPROM, jika valid transfer to HoldingRegs

  transferFromProcessToHoldingRegsArdu(); //Transfer dari Relay parameter ke HoldingRegsArdu
  equelizingHostData(); //Menyamakan awal HoldingRegsHost dan HoldingRegsArdu

  SequenceUtama.setSetingMiliSecond(500); //Setup Sequence utama 500ms

//semua terkait dengan menu
  setupMenuReleComb(); //lihat file MenuReleComb
  subMenuInfo(); //Tampilkan display info Relecomb ke LCD

//terkait RTC dan SD Card
  // Setup SD Card
  SD.begin();
  ///*
  //RTC DS1307
  rtc.begin();
  //*/
  /*
  //Software RTC
    rtc.begin(DateTime(F(__DATE__), F(__TIME__)));
  */
}

void loop() {
```

```
  // put your main code here, to run repeatedly:
  // Running on 20 mili Second
  if (SequenceUtama.isMiliSecondEvent()) { //Dijalankan tiap 500 ms
    updateCommandFromHost(); //Check data dari Host dulu dan transfer
sebagian ke operasi
    transferFromProcessToHoldingRegsArdu(); //Semua Proses disimpan
ke HoldingRegs
    mainMenuRele(); //Menampilkan menu LCD
  }
  if (SequenceUtama.isASecondEvent()){ //Dijalankan setiap detik

monitoringArusDanTegangan.calcVI(CROSSING_MAX,TIME_OUT_
MAX); // Calculate all. No.of half wavelengths (crossings), time-out
    digitalWrite( PIN_LED_LIVE, digitalRead(PIN_LED_LIVE) ^ 1 );
//Jalankan LED pin 13
    SaatIni = rtc.now(); //Deklarasi waktu saat ini
    DataLog(); //Simpan datalogger ke SD Card
  }
  modbus_update(); //Deklarasi protokol Modbus
}

//Lokasi penempatan prosedur
void arduInterfaceSetup(){
  lcd.begin(LCD_COLUMNS, LCD_ROWS);
  init_phi_prompt(&lcd,keypads,function_keys, LCD_COLUMNS,
LCD_ROWS, '~'); // Supply the liquid crystal object, input keypads, and
function key names. Also supply the column and row of the lcd, and
indicator as '>'. You can also use '\x7e', which is a right arrow.
}

void setupDefault(){
  /*setup Relay Protection Short Circuit*/
  RelayShortCirt.setTimeChar(DEF_TIME);//Time Characteristic
  RelayShortCirt.setTimeDef(500);
  RelayShortCirt.setSetting(3.0);//set setting value
  /*setup Relay Protection Over Load (Tambahan 20/05/2017)*/
  RelayOverLoad.setTimeChar(INVS_STD);//Time Characteristic
  RelayOverLoad.setTimeDef(1500);
  RelayOverLoad.setSetting(1.5);//set setting value
```

```
}

void setupRelayShortCirt(){
  //setup Relay Protection Short Circuit
  RelayShortCirt.setPinTest(PIN_TRIP_SWITCH);
  RelayShortCirt.setPinReset(PIN_RESET_SWITCH);
  RelayShortCirt.setPinRL(PIN_RELAY_OUT);
  RelayShortCirt.setActionChar(OVR_ACTION);
  RelayShortCirt.setActive(true);
}

void setupRelayOverLoad(){
  //setup Relay Protection Over Load
  RelayOverLoad.setPinTest(PIN_TRIP_SWITCH);
  RelayOverLoad.setPinReset(PIN_RESET_SWITCH);
  RelayOverLoad.setPinRL(PIN_RELAY_OUT);
  RelayOverLoad.setActionChar(OVR_ACTION);
  RelayOverLoad.setActive(true);
}

 void equelizingHostData(){
/*
//For Short Circuit
  ARDU_SH_SP,  //index 1
  ARDU_SH_PV,  //index 2
  ARDU_SH_ACTION,  //index 3
  ARDU_SH_TCHAR,  //index 4
  ARDU_SH_TSET,  //index 5
  ARDU_SH_TMS,  //index 6
  ARDU_SH_TACC,  //index 7
//For Over Load
  ARDU_OL_SP,  //index 8
  ARDU_OL_PV,  //index 9
  ARDU_OL_ACTION,  //index 10
  ARDU_OL_TCHAR,  //index 11
  ARDU_OL_TSET,  //index 12
  ARDU_OL_TMS,  //index 13
  ARDU_OL_TACC,  //index 14
//for setting data Short Circuit
```

```
  HOST_SH_SP,  //index 20
  HOST_SH_ACTION,  //index 21
  HOST_SH_TCHAR,  //index 22
  HOST_SH_TSET,  //index 23
  HOST_SH_TMS,  //index 24
//for setting data Over Load
  HOST_OL_SP,  //index 25
  HOST_OL_ACTION,  //index 26
  HOST_OL_TCHAR,  //index 27
  HOST_OL_TSET,  //index 28
  HOST_OL_TMS,  //index 29
*/
//Short Circuit = Host --> Ardu
  holdingRegs[HOST_SH_SP] = holdingRegs[ARDU_SH_SP];
  holdingRegs[HOST_SH_ACTION]                                    =
holdingRegs[ARDU_SH_ACTION];
  holdingRegs[HOST_SH_TCHAR]                                     =
holdingRegs[ARDU_SH_TCHAR];
  holdingRegs[HOST_SH_TSET] = holdingRegs[ARDU_SH_TSET];
  holdingRegs[HOST_SH_TMS] = holdingRegs[ARDU_SH_TMS];
//Over Load = Host --> Ardu
  holdingRegs[HOST_OL_SP] = holdingRegs[ARDU_OL_SP];
  holdingRegs[HOST_OL_ACTION]                                    =
holdingRegs[ARDU_OL_ACTION];
  holdingRegs[HOST_OL_TCHAR]                                     =
holdingRegs[ARDU_OL_TCHAR];
  holdingRegs[HOST_OL_TSET] = holdingRegs[ARDU_OL_TSET];
  holdingRegs[HOST_OL_TMS] = holdingRegs[ARDU_OL_TMS];
}

// Interrupt is called once a millisecond,
SIGNAL(TIMER0_COMPA_vect)
{
  unsigned long currMilli,  delta;
  currMilli  = millis();
  delta = currMilli  - prevMilli;
  SequenceUtama.execute();
  itrNmbr++;
  if (delta >= A20MILLISECOND){
```

```
  //Jalankan Fungsi Proteksi
  RelayShortCirt.execute(monitoringArusDanTegangan.Irms, millis());
//Beri nilai dan jalankan (Short Circuit)
  RelayOverLoad.execute(monitoringArusDanTegangan.Irms, millis());
//Beri nilai dan jalankan (Over Load)
  if (RelayShortCirt.getState() >= STATUS_TRIP) {
    //Tulis Event Ke EEPROM
    MenulisEventKeEEPROM(SHORT_CIRCUIT); //Simpan data
gangguan ke EEPROM
    sudahTulisEvent = false;
  }
  else if (RelayOverLoad.getState() >= STATUS_TRIP) {
    //Tulis Event Ke EEPROM
    MenulisEventKeEEPROM(OVER_LOAD);// Simpan data gangguan
ke EEPROM
    sudahTulisEvent = false;
  }
  else sudahTulisEvent = true;
    itrNmbr=0;
    prevMilli = currMilli;
  }
}

void SetupEmonlib()
{
  monitoringArusDanTegangan.voltage(1, 233, 1.7); // Voltage: input
pin, calibration, phase_shift (RL1)
  //monitoringArusDanTegangan.voltage(1, 250, 1.7); // Voltage: input
pin, calibration, phase_shift (RL2)
  //monitoringArusDanTegangan.current(3, 10); // Current: input pin,
calibration. (default)
  monitoringArusDanTegangan.current(3,
9.68081395348837209302325581395 35); // Current: input pin,
calibration. (RL1)
  //monitoringArusDanTegangan.current(3,
10.48023901876506971380647866 6527); // Current: input pin,
calibration. (RL2)
}
```

```
void DataLog(){
char Time[20];  //Pendeklarasian Waktu
char Date[20];  //Pendeklarasian Tanggal
 sprintf(Time, "%02d:%02d:%02d", SaatIni.hour(), SaatIni.minute(),
SaatIni.second());
 sprintf(Date, "%02d/%02d/%02d", SaatIni.day(), SaatIni.month(),
SaatIni.year());
 myFile = SD.open("MMC.txt", FILE_WRITE); //membuka file dan
menulis isi filenya
 if (myFile) {
  myFile.print(Date); // print ke file yang ada di micro SD
  myFile.print(",");
  myFile.print(Time);
  myFile.print(",");
  myFile.print(monitoringArusDanTegangan.Irms);
  myFile.print(",");
  myFile.println(monitoringArusDanTegangan.Vrms);
  myFile.close(); // tutup file
 }
 // jika file tidak dapat dibuka, print error.
 else {
  Serial.println("error opening test.txt");
 }
}

void MenulisEventKeEEPROM(byte EventVal){
 if (sudahTulisEvent == true){
  byte JumlahEvent =
EEPROM.read(ALAMAT_EEPROM_AWAL_EVENT-1);
//Mengetahui jumlah event terakhir
  if (JumlahEvent == 0xFF)JumlahEvent = 1; //Antisipasi jika Arduino
belum pernah menulis event
  JumlahEvent++;
  if (JumlahEvent >= JUMLAH_EVENT_MAX) JumlahEvent = 1;
  int AlamatEvent = ALAMAT_EEPROM_AWAL_EVENT +
(JumlahEvent-1)*sizeof(DataEvent);
  MengisiDataEvent(JumlahEvent, EventVal);
  EEPROM.put(AlamatEvent, DataTulis);
```

```
    EEPROM.write(ALAMAT_EEPROM_AWAL_EVENT-
1,JumlahEvent);
    transferEventToHoldingRegs(DataTulis); //Update HoldingRegs
   }
}

void readEventFromEEPROM(){
  byte                    tempEventNbr                    =
EEPROM.read(ALAMAT_EEPROM_AWAL_EVENT-1);
  if      ((tempEventNbr      >      0)      &&      (tempEventNbr      <=
JUMLAH_EVENT_MAX)){

transferEventToHoldingRegs(MembacaEventDariEEPROM(tempEvent
Nbr));
  }
}
DataEvent MembacaEventDariEEPROM(byte  NomorEvent){
  DataEvent tempEvent;
  int    AlamatEvent    =    ALAMAT_EEPROM_AWAL_EVENT    +
(NomorEvent-1)*sizeof(DataEvent);
  EEPROM.get(AlamatEvent,  tempEvent);
  return tempEvent;
}

void MengisiDataEvent(byte NomorEvent, byte EventVal){
  /*Status Short Circuit  dan Over Load*/
  if (EventVal  ==  SHORT_CIRCUIT){
    DataTulis.Arus = RelayShortCirt.getValue();
    DataTulis.Status = RelayShortCirt.getState();
  }
  if (EventVal  ==  OVER_LOAD){
    DataTulis.Arus = RelayOverLoad.getValue();
    DataTulis.Status = RelayOverLoad.getState();
  }
  DataTulis.EventType = EventVal;
  DataTulis.Tegangan = monitoringArusDanTegangan.Vrms;
  DataTulis.NomorEvent  = NomorEvent;
  DataTulis.Detik  =SaatIni.second() ;
  DataTulis.Menit = SaatIni.minute();
```

```
  DataTulis.Jam = SaatIni.hour();
  DataTulis.Tanggal = SaatIni.day();
  DataTulis.Bulan = SaatIni.month();
  DataTulis.Tahun = SaatIni.year();
}

//Untuk Menentukan Alamat Komunikasi
byte getSlaveId(){
  byte _slaveID=0;
  if (digitalRead(SLAVE_ADDR_PIN0))_slaveID=_slaveID |
B00000001;
  return (_slaveID+1);
}
//Setup Modbus, Addressing dan konfigurasi
void setupModbus(){
  pinMode(SLAVE_ADDR_PIN0, INPUT);//untuk alamat Slave

  byte slaveId = getSlaveId();
  modbus_configure(&Serial1, BAUD_RATE, SERIAL_8N2, slaveId,
PIN_TX_RX_ENABLE, HOLDING_REGS_SIZE, holdingRegs);
}

/*HostComm*/
void transferFromProcessToHoldingRegsArdu(){
  holdingRegs[ARDU_ID] = getSlaveId();
//Proses --> HoldingRegs (Short Circuit)
  holdingRegs[ARDU_SH_PV] =
RelayShortCirt.getValue()*WORD_MAX/MAX_PV;
  holdingRegs[ARDU_SH_STATE] = RelayShortCirt.getState();
  holdingRegs[ARDU_SH_SP] =
RelayShortCirt.getSetting()*WORD_MAX/MAX_PV;
  holdingRegs[ARDU_SH_TCHAR] = RelayShortCirt.getTimeChar();
  holdingRegs[ARDU_SH_TSET] = RelayShortCirt.getTimeDef();
  holdingRegs[ARDU_SH_TMS] = RelayShortCirt.getTMS();
  holdingRegs[ARDU_SH_TACC] = RelayShortCirt.getTimeACC();
  holdingRegs[ARDU_SH_ACTION] = RelayShortCirt.getActionChar();
//Proses --> HoldingRegs (Over Load)
  holdingRegs[ARDU_OL_PV] =
RelayOverLoad.getValue()*WORD_MAX/MAX_PV;
```

```
  holdingRegs[ARDU_OL_STATE]  = RelayOverLoad.getState();
  holdingRegs[ARDU_OL_SP]                                          =
RelayOverLoad.getSetting()*WORD_MAX/MAX_PV;
  holdingRegs[ARDU_OL_TCHAR]  = RelayOverLoad.getTimeChar();
  holdingRegs[ARDU_OL_TSET]  = RelayOverLoad.getTimeDef();
  holdingRegs[ARDU_OL_TMS]  = RelayOverLoad.getTMS();
  holdingRegs[ARDU_OL_TACC]  = RelayOverLoad.getTimeACC();
  holdingRegs[ARDU_OL_ACTION]                                      =
RelayOverLoad.getActionChar();
  //Untuk jumlah event
  holdingRegs[ARDU_EVENT_NBR]                                      =
EEPROM.read(ALAMAT_EEPROM_AWAL_EVENT-1);
//Mengetahui jumlah event terakhir
}

void updateCommandFromHost(){
  DataEvent tempEvent;
  word startAddr, lenghtAddr;
  byte OprStatus,LastOprStatus;
  byte HostCommand  = holdingRegs[HOST_OPR_MODE];

  byte valStatus = RelayShortCirt.getState();
  if      (valStatus      <      RelayOverLoad.getState())valStatus      =
RelayOverLoad.getState(); //Ambil yang tertinggi

//Operational EEPROM
  OprStatus = 0;
  LastOprStatus = OprStatus;
  switch (HostCommand)
  {
    case TRIP_REMOTE:
      OprStatus++;
      RelayShortCirt.setState(STATUS_TEST_REMOTE);
      OprStatus++;
      RelayOverLoad.setState(STATUS_TEST_REMOTE);
      OprStatus++;
    break;
    case RESET_REMOTE:
      //Reset tidak bisa dioperasikan jika LocalTrip  aktif
```

```
    OprStatus++;
    valStatus = valStatus & STATUS_TEST_LOCAL; //Ambil bit test
lokal
    if (valStatus != STATUS_TEST_LOCAL){
      OprStatus++;
      RelayShortCirt.setReset(true);
      OprStatus++;
      RelayOverLoad.setReset(true);
      OprStatus++;
    }
  break;
  case CHANGE_SET://
    OprStatus++;
    if (holdingRegs[HOST_SET_VERSION] >= 0){ //Check dan
jalankan jika ada update Setting dari Host/Master
      OprStatus++;
      if (DataReleShortCircuit.Version !=
holdingRegs[HOST_SET_VERSION]){ //Check dan jalankan jika ada
update Setting dari Host/Master
        OprStatus++;
        OprStatus = OprStatus +
transferFromHostHoldingRegsToDataRelay(); //transfer to dataRelay,
process and write to EEPROM
      }
    }
  break;
  case DEFAULT_SET: //Kembali ke Setting Default
    OprStatus++;
    if (holdingRegs[HOST_SET_VERSION] > 0 ){
      startAddr = ALAMAT_EEPROM_AWAL_DATA;
      lenghtAddr = 2 * sizeof(DataSettingRelay); //Setting ShortCircuit
dan OverCurrent
      OprStatus++;
      ClearEEPROM(startAddr,lenghtAddr);
      OprStatus++;
      setupDefault(); //Setting Default
      OprStatus++;
      transferFromProcessToHoldingRegsArdu(); //Transfer dari Relay
parameter ke HoldingRegsArdu
```

```
      OprStatus++;
      equelizingHostData(); //Menyamakan awal HoldingRegsHost dan
HoldingRegsArdu
      holdingRegs[HOST_SET_VERSION] = 0;
      OprStatus++;
     }
   break;
   case READ_EVENT: //Baca Event
    OprStatus++;
    if (holdingRegs[HOST_EVENT_WILL_READ] != 0 ){
    OprStatus++;
     tempEvent                                                    =
MembacaEventDariEEPROM(holdingRegs[HOST_EVENT_WILL_RE
AD]);
      OprStatus++;
      if          (holdingRegs[HOST_EVENT_WILL_READ]          ==
tempEvent.NomorEvent){ //Valid event number
       OprStatus++;
       transferEventToHoldingRegs(tempEvent);
       OprStatus++;
      }
    }
    //Akhir dari baca All Event
    if ((holdingRegs[HOST_EVENT_WILL_READ] == 0 ) &&
(holdingRegs[HOST_EVENT_NUMBER] > 0 )){
      OprStatus++;
       tempEvent                                                   =
MembacaEventDariEEPROM(holdingRegs[HOST_EVENT_NUMBER
]);
       OprStatus++;
       if          (holdingRegs[HOST_EVENT_NUMBER]          ==
tempEvent.NomorEvent){ //Valid event number
         OprStatus++;
         transferEventToHoldingRegs(tempEvent);
         OprStatus++;
        }
    }
   break;
   case CLEAR_EVENT: //Hapus semua event
```

```
      OprStatus++;
      if (holdingRegs[HOST_EVENT_NUMBER]  > 0 ){
        startAddr = ALAMAT_EEPROM_AWAL_EVENT  - 1;
        lenghtAddr  =  JUMLAH_EVENT_MAX  *  sizeof(DataEvent);
//Event histories
        OprStatus++;
        ClearEEPROM(startAddr,lenghtAddr);
        holdingRegs[HOST_EVENT_NUMBER]  = 0;
        OprStatus++;
       }
     break;
     default:
     break;
   }
   if  (LastOprStatus  <  OprStatus)LastOprStatus  =  OprStatus;  //Ambil
status tertinggi
//Operation has been completed with last status recorded
   if (HostCommand  > NO_OPERATION)  {
     if (LastOprStatus >= OprStatus){
       holdingRegs[HOST_OPR_STATUS]  = LastOprStatus;
       holdingRegs[LAST_OPR_MODE]  = HostCommand;
       holdingRegs[HOST_OPR_MODE]  = NO_OPERATION;
     }
   }
}

void transferEventToHoldingRegs(DataEvent tempEvent){
    holdingRegs[HOST_EVENT_NUMBER]=tempEvent.NomorEvent;
    holdingRegs[HOST_EVENT_TAHUN]=tempEvent.Tahun;
    holdingRegs[HOST_EVENT_BULAN]=tempEvent.Bulan;
    holdingRegs[HOST_EVENT_TGL]=tempEvent.Tanggal;
    holdingRegs[HOST_EVENT_JAM]=tempEvent.Jam;
    holdingRegs[HOST_EVENT_MENIT]=tempEvent.Menit;
    holdingRegs[HOST_EVENT_DETIK]=tempEvent.Detik;

holdingRegs[HOST_EVENT_ARUS]=tempEvent.Arus*WORD_MAX/
MAX_PV;
```

```
holdingRegs[HOST_EVENT_TEG]=tempEvent.Tegangan*WORD_M
AX/MAX_TEG;
   holdingRegs[HOST_EVENT_STATUS]=tempEvent.Status;
   holdingRegs[HOST_EVENT_TYPE]=tempEvent.EventType;
}

void transferFromDataRelayToRelayShortCirt(){
  RelayShortCirt.setTimeChar(DataReleShortCircuit.TChar);
  RelayShortCirt.setActionChar(DataReleShortCircuit.Action);
  RelayShortCirt.setSetting(DataReleShortCircuit.SetPoint);
  RelayShortCirt.setTMS(DataReleShortCircuit.TMS);
  RelayShortCirt.setTimeDef(DataReleShortCircuit.TSet);
}

void transferFromDataRelayToRelayOverLoad(){
  RelayOverLoad.setTimeChar(DataReleOverLoad.TChar);
  RelayOverLoad.setActionChar(DataReleOverLoad.Action);
  RelayOverLoad.setSetting(DataReleOverLoad.SetPoint);
  RelayOverLoad.setTMS(DataReleOverLoad.TMS);
  RelayOverLoad.setTimeDef(DataReleOverLoad.TSet);
}

byte transferFromHostHoldingRegsToDataRelay(){
  byte stepFunction = 1;
//Data setting RelayShorCircuit
  DataReleShortCircuit.TChar  = holdingRegs[HOST_SH_TCHAR];
  DataReleShortCircuit.Action  = holdingRegs[HOST_SH_ACTION];
  DataReleShortCircuit.SetPoint                                    =
holdingRegs[HOST_SH_SP]*MAX_PV/WORD_MAX;
  DataReleShortCircuit.TMS  = holdingRegs[HOST_SH_TMS];
  DataReleShortCircuit.TSet  = holdingRegs[HOST_SH_TSET];

  transferFromDataRelayToRelayShortCirt();
  stepFunction++;//increase a step

//Data setting RelayOverLoad  22/05/2017
  DataReleOverLoad.TChar  = holdingRegs[HOST_OL_TCHAR];
  DataReleOverLoad.Action  = holdingRegs[HOST_OL_ACTION];
```

```
  DataReleOverLoad.SetPoint                                    =
holdingRegs[HOST_OL_SP]*MAX_PV/WORD_MAX;
  DataReleOverLoad.TMS  = holdingRegs[HOST_OL_TMS];
  DataReleOverLoad.TSet  = holdingRegs[HOST_OL_TSET];

  transferFromDataRelayToRelayOverLoad();
  stepFunction++; //increase a step

//penambahan Version  dan waktu
  if          (holdingRegs[HOST_SET_VERSION]          >=          0xFF)
holdingRegs[HOST_SET_VERSION]  = 1; //Kembali ke 1/roll over
  DataReleShortCircuit.Version  = holdingRegs[HOST_SET_VERSION]
; //Di-update dari Host
  DataReleShortCircuit.Detik  =SaatIni.second() ;
  DataReleShortCircuit.Menit  = SaatIni.minute();
  DataReleShortCircuit.Jam  = SaatIni.hour();
  DataReleShortCircuit.Tanggal  = SaatIni.day();
  DataReleShortCircuit.Bulan  = SaatIni.month();
  DataReleShortCircuit.Tahun  = SaatIni.year();
  stepFunction++; //increase a step

//Data setting RelayOverLoad
  DataReleOverLoad.Version   =  holdingRegs[HOST_SET_VERSION]
;//di-update dari Host
  DataReleOverLoad.Detik  =SaatIni.second() ;
  DataReleOverLoad.Menit  = SaatIni.minute();
  DataReleOverLoad.Jam  = SaatIni.hour();
  DataReleOverLoad.Tanggal  = SaatIni.day();
  DataReleOverLoad.Bulan  = SaatIni.month();
  DataReleOverLoad.Tahun  = SaatIni.year();

  stepFunction++; //increase a step

  EEPROM.put(ALAMAT_EEPROM_AWAL_DATA,
DataReleShortCircuit);
  stepFunction++; //increase a step
  EEPROM.put(ALAMAT_EEPROM_AWAL_DATA                           +
sizeof(DataSettingRelay),  DataReleOverLoad);
  stepFunction++; //increase a step
```

```
  return stepFunction;
}

void updateDataRelayFromEEPROM(){
  boolean DataRelayValid = true;
  EEPROM.get(ALAMAT_EEPROM_AWAL_DATA,
DataReleShortCircuit);
  EEPROM.get(ALAMAT_EEPROM_AWAL_DATA +
sizeof(DataSettingRelay), DataReleOverLoad);

//check setting ShortCircuit Valid
  if ((DataReleShortCircuit.Version <= 0)|(DataReleShortCircuit.Version
>= 0xFF)){
    DataRelayValid = false;
    DataReleShortCircuit.Version = 0;
    DataReleOverLoad.Version = 0;
  }
//check setting OverLoad Valid
  if ((DataReleOverLoad.Version <= 0)|(DataReleOverLoad.Version >=
0xFF)){
    DataRelayValid = false;
    DataReleShortCircuit.Version = 0;
    DataReleOverLoad.Version = 0;
  }
  if (DataRelayValid){
    transferFromDataRelayToRelayShortCirt();
    transferFromDataRelayToRelayOverLoad();

    holdingRegs[HOST_SET_VERSION] =
DataReleShortCircuit.Version;
    holdingRegs[HOST_SET_TAHUN] = DataReleShortCircuit.Tahun;
    holdingRegs[HOST_SET_BULAN] = DataReleShortCircuit.Bulan;
    holdingRegs[HOST_SET_TGL] = DataReleShortCircuit.Tanggal;
    holdingRegs[HOST_SET_JAM] = DataReleShortCircuit.Jam;
    holdingRegs[HOST_SET_MENIT] = DataReleShortCircuit.Menit;
    holdingRegs[HOST_SET_DETIK] = DataReleShortCircuit.Detik;
  }
}
void ClearEEPROM(word startAddr, word lenghtAddr){
```

```
  word maxEEPROMaddr  = startAddr+ lenghtAddr;
  if (maxEEPROMaddr  < EEPROM.length()){
    for (int i = 0 ; i < lenghtAddr; i++)  {
      EEPROM.write(i+startAddr,  0);
    }
  }
}

/*MenuReleComb*/
/** \file
Relecomb menu sbb:
Menu Utama:  -> Index 0
  Rele1 -> Trip-Reset  Index -> Index 10
    Test Trip -> Index 14
    Reset -> Index 15
  DisPlay -> Index 20
  Status -> Index 30
  Info -> Index 40
*/
const char ReleCombMn0[]  PROGMEM="Trip-Reset";
const char ReleCombMn1[]  PROGMEM="Display";
const char ReleCombMn2[]  PROGMEM="Status";
const char ReleCombMn3[]  PROGMEM="Info";
const char* const ReleCombMnItems[]  PROGMEM  = {ReleCombMn0,
ReleCombMn1,  ReleCombMn2,  ReleCombMn3};
phi_prompt_struct mainMenu; //This structure stores the main  menu.

//This program is the main menu. It handles inputs from the keys, updates
the menu or executes a certain menu function accordingly.
void mainMenuRele(){
   switch (menuId.mainMenu)  //See which  menu item  is selected and
execute that correS_Pond function
   {
     case MAIN_MENU_IDX:
       menuReleComb();
     break;
     case TRIP_RESET_MENU_IDX:
       subMenuTripReset("<<-Trip-Reset->>"); //lihat file subMenuRele
     break;
```

```
    case DISPLAY_MENU_IDX:
      subMenuDisPlay(); //Menampilkan tegangan dan arus setting/actual
    break;
    case STATUS_MENU_IDX:
      subMenuStatus(); //Menampilkan tgl,bln,thn jam:mm:ss  dan status
relay
    break;
    case INFO_MENU_IDX:
      subMenuInfo();
    break;
    default:
    break;
   }
}

void setupMenuReleComb(){
  mainMenu.ptr.list=(char**)&ReleCombMnItems;  // Assign the list to
the pointer
  mainMenu.low.i=0;  // Default item highlighted on the list
  mainMenu.high.i=2;  // Last item of the list is size of the list - 1.
  mainMenu.width=LCD_COLUMNS-
((global_style&phi_prompt_arrow_dot)!=0)-
((global_style&phi_prompt_scroll_bar)!=0);  // Auto fit the size of the list
to the screen. Length in characters of the longest list item.
  mainMenu.step.c_arr[0]=LCD_ROWS-1;  // rows to auto fit entire
screen
  mainMenu.step.c_arr[1]=1;  // one col list
  mainMenu.step.c_arr[2]=0;  // y for additional feature such as an index
  mainMenu.step.c_arr[3]=LCD_COLUMNS-4-
((global_style&phi_prompt_index_list)!=0);  // x for additional feature
such as an index
  mainMenu.col=0; // DisPlay menu at column 0
  mainMenu.row=1;  // DisPlay menu at row 1
  mainMenu.option=global_style; // Option 0, disPlay classic list, option
1, disPlay 2X2 list, option 2, disPlay list with index, option 3, disPlay list
with index2.
}

void menuReleComb()
```

```
{
  int menu_pointer_1=0; // This stores the menu choice the user made.
  lcd.clear();  // Refresh menu if a button has been pushed
  center_text("Main Menu");//Menu Title

  select_list(&mainMenu); // Use the select_list to ask the user to select
an item of the list, that is a menu item from your menu.
  menu_pointer_1=mainMenu.low.i;  // Get the selected item number and
store it in the menu pointer.
  switch (menu_pointer_1) // See which menu item is selected and execute
that correS_Pond function
  {
   case 0:
     menuId.mainMenu = TRIP_RESET_MENU_IDX;
     menuId.subMenu = 1;
   break;
   case 1:
     menuId.mainMenu = DISPLAY_MENU_IDX;
     menuId.subMenu = 1;
   break;
   case 2:
     menuId.mainMenu = STATUS_MENU_IDX;
     menuId.subMenu = 1;
   break;
   case 3:
     menuId.mainMenu = INFO_MENU_IDX;
     menuId.subMenu = 1;
   break;
   default:
   break;
  }
}

void subMenuDisPlay(){
  byte currentKey;
  String Sval;
  lcd.clear();  // Refresh menu if a button has been pushed

//Tampilkan  pada baris ke 1
```

```
  Sval =  String("Set:");
  Sval =  String(Sval + RelayShortCirt.getSetting()); //EmonLib  Arus SC
  Sval =  String(Sval + "/");
  Sval =  String(Sval + RelayOverLoad.getSetting()); //EmonLib  Arus OL
  Sval =  String(Sval + " A");
  lcd.setCursor(0,0);//posisikan kursor pada baris 1 kolom 1
  lcd.print(Sval);

//Tampilkan  Nilai  Tegangan pada baris ke 2
  Sval =  String("V/I:");
  Sval =  String(Sval + monitoringArusDanTegangan.Vrms);  //EmonLib
Tegangan
  Sval =  String(Sval + "/");
  Sval =  String(Sval + RelayShortCirt.getValue());   //EmonLib  Arus
  lcd.setCursor(0,1);  //Posisikan kursor pada baris 2 kolom  1
  lcd.print(Sval);

  currentKey = analogKeypad.getKey(); // Use phi_keypads object to
access the keypad
  switch (currentKey) // See which menu item is selected and execute that
correS_Pond function
  {
    case 'S':
      menuId.mainMenu  = MAIN_MENU_IDX;
      menuId.subMenu = 1;
      return;
    break;
    default:
    break;
  }
}

void subMenuStatus(){
  // menu untuk menampilkan  tgl dan waktu pada baris pertama
  // status relay pada baris kedua
  byte currentKey;
  String  Sval;
  lcd.clear();   // Refresh  menu  if a button has been pushed
```

```
//Tampilkan pada baris ke 1
  Sval = String(SaatIni.day());//ambil tanggal
  Sval = String(Sval + "/");
  Sval = String(Sval + SaatIni.month());//ambil bulan
  Sval = String(Sval + "/");
  Sval = String(Sval + (SaatIni.year()-2000));////ambil tahun
  Sval = String(Sval + " ");
  Sval = String(Sval + SaatIni.hour());////ambil jam
  Sval = String(Sval + ":");
  Sval = String(Sval + SaatIni.minute());////ambil menit
  Sval = String(Sval + ":");
  Sval = String(Sval + SaatIni.second());////ambil detik

  lcd.setCursor(0,0);
  lcd.print(Sval);

//Tampilkan status relay pada baris ke 2
  lcd.setCursor(0,1);
  lcd.print(statusRelay());
  currentKey = analogKeypad.getKey(); // Use phi_keypads object to
access the keypad
  switch (currentKey) // See which menu item is selected and execute that
correS_Pond function
  {
    case 'S':
      menuId.mainMenu = MAIN_MENU_IDX;
      menuId.subMenu = 1;
      return;
    break;
    default:
    break;
  }
}

String statusRelay(){
  byte _state;
  String Sval;
  _state = RelayShortCirt.getState();
```

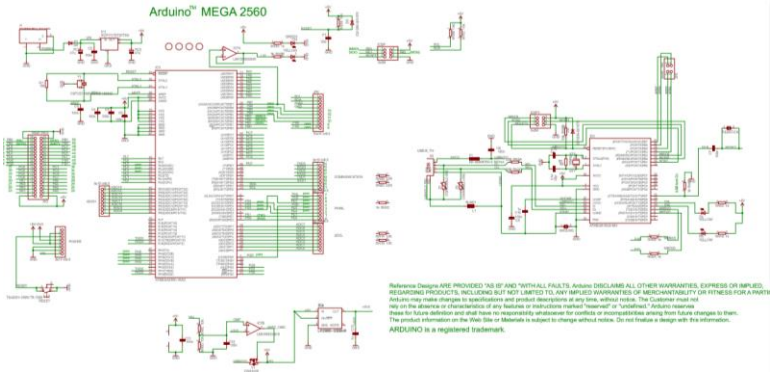
```
  if (_state < RelayOverLoad.getState())_state = RelayOverLoad.getState(); //ambil nilai status tertinggi
  Sval = String("STS:");
  if (_state == STATUS_OK)return (Sval = String(Sval + "OK"));
  else if (_state >= STATUS_TRIP){
    Sval = String(Sval + "TRP->");
    _state = _state - STATUS_TRIP;
  }
  if (_state > STATUS_OK){//Trip atau belum trip dengan beberapa status
    //check status Relay
    switch (_state){
      case STATUS_OVER:
        return String(Sval + "OVR");
      break;
      case STATUS_UNDER:
        return String(Sval + "UDR");
      break;
      case EQL_ACTION:
        return String(Sval + "EQL");
      break;
      case STATUS_TEST_LOCAL:
        return String(Sval + "LOCAL");
      break;
      case STATUS_TEST_KEY:
        return String(Sval + "KEYPAD");
      break;
      case STATUS_TEST_REMOTE:
        return String(Sval + "REMOTE");
      break;
      default:
      break;
    }
  }
}

void subMenuInfo(){
char infoMsg[]="by: Abi and Sindhu ";
char buffer[15];
lcd.clear();
```

```
lcd.noBlink();
center_text("Protection Relay"); // display judul
for (byte i=0;i<strlen(infoMsg);i++)
  {
   scroll_text(infoMsg,buffer,14,i-14);
   lcd.setCursor(1,1);
   lcd.print(buffer);
   wait_on_escape(300);
  }
 menuId.mainMenu = DISPLAY_MENU_IDX;
 menuId.subMenu = 1;
}

void subMenuTripReset(char* judulMenu)
{
 byte currentKey;
 lcd.clear();  // Refresh menu if a button has been pushed
 center_text(judulMenu);//Menu Title
 currentKey = analogKeypad.getKey(); // Use phi_keypads object to
access the keypad
 switch (currentKey) // See which menu item is selected and execute that
correS_Pond function
 {
  case 'U':
    //menuId.mainMenu = TRIP_RESET_MENU_IDX;
    menuId.subMenu--;
    if (menuId.subMenu < TRIP_MENU_IDX) menuId.subMenu =
BACK_MENU_IDX;  //roll over to Back_maneu
  break;
  case 'D':
    menuId.subMenu++;
    if (menuId.subMenu > BACK_MENU_IDX) menuId.subMenu =
TRIP_MENU_IDX;  //roll over to Trip_menu
  break;
  case 'S':
    if (menuId.subMenu == TRIP_MENU_IDX){
      menuId.mainMenu = 0;
      RelayShortCirt.setState(STATUS_TEST_KEY);  //Test trip dari
keypad (SC)
```

```
      RelayOverLoad.setState(STATUS_TEST_KEY);   //Test trip dari
keypad (OL)
       return;//ini untuk test trip
      }
      if (menuId.subMenu ==  RESET_MENU_IDX){
       menuId.mainMenu = 0;
       RelayShortCirt.setReset(true); //Reset dari keypad (SC)
       RelayOverLoad.setReset(true); //Reset dari keypad (OL)
       return;//ini untuk Reset
      }
      if (menuId.subMenu ==  BACK_MENU_IDX){
       menuId.mainMenu = 0;
       return;
      }
    break;
    default:
    break;
  }
  lcd.clear();  // Refresh menu if a button has been pushed
  center_text(judulMenu);//Menu Title
  DisplayRelayLCD();
}

void DisplayRelayLCD(){
  String Sval;
  byte _timeChar;
  if (menuId.subMenu == TRIP_MENU_IDX){
    Sval =  String("Test Trip");
  }
  if (menuId.subMenu == RESET_MENU_IDX){
    Sval =  String("Reset");
  }
  if (menuId.subMenu == BACK_MENU_IDX){
    Sval =  String("Back To Main Menu");
  }
  lcd.setCursor(0,1);
  lcd.print(Sval);
}
```

LAMPIRAN B

B.1 DATASHEET ARDUINO UNO



Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board

Power

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I²C: 20 (SDA) and 21 (SCL).** Support I²C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I²C pins on the Duemilanove.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Mega has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila. **Please note that I²C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).**

B.2. DATASHEET YHDC SCT-013

Product Specification

Date:2015-8-7

Product Name	Current transformer	Model	SCT013-010

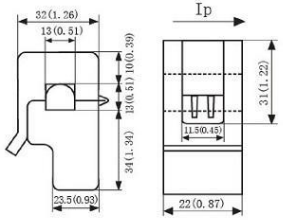
Characteristics:Openning size 13mm*13mm,1m leading wire, standard Φ3.5 three-core plug output. Have two kinds of output type: Current output type and voltage output type.
Purpose： Used for current measurement, monitor and protection for AC motor,lighting equipment, air compressor etc

Technical Data

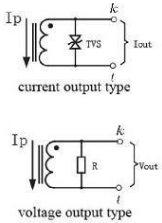
I_{PN}	Rated input	0-10A
I_{PM}	Max. detection input	
I_{OUT}	Rated output	0-1V
X	Accuracy	±1%
ε_L	Linearity	≤0.2%
N	Turns ratio	1:1800
Φ	Phase shift	
R_L	Max.Sampling resistance	
V_{PN}	Work voltage	660V
f	Work frequency	50-1KHz
T_A	Operating temperature	-25..+70℃
T_S	Storage temperature	-40..+85℃
Vd	Dielectric strength, 50 Hz, 1 min	3KV

Fire resistance	UL94-V0
Material of core	Ferrite
Mounting type	Suspension
Weight	55g

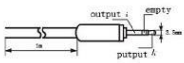
Dimension (mm(in). 1 mm= 0.0394 inch)



Front view　　Side view



Schematic diagram



Standard three-core pligs schematic diagram

B.3. DATASHEET DS1307

DALLAS SEMICONDUCTOR MAXIM

DS1307
64 x 8, Serial, I²C Real-Time Clock

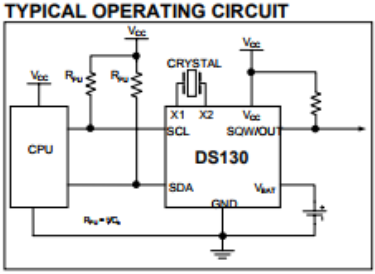
GENERAL DESCRIPTION

The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I²C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply.

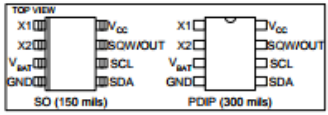
FEATURES

- Real-Time Clock (RTC) Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the week, and Year with Leap-Year Compensation Valid Up to 2100
- 56-Byte, Battery-Backed, General-Purpose RAM with Unlimited Writes
- I²C Serial Interface
- Programmable Square-Wave Output Signal
- Automatic Power-Fail Detect and Switch Circuitry
- Consumes Less than 500nA in Battery-Backup Mode with Oscillator Running
- Optional Industrial Temperature Range: -40°C to +85°C
- Available in 8-Pin Plastic DIP or SO
- Underwriters Laboratories (UL) Recognized

TYPICAL OPERATING CIRCUIT



PIN CONFIGURATIONS



ORDERING INFORMATION

PART	TEMP RANGE	VOLTAGE (V)	PIN-PACKAGE	TOP MARK*
DS1307+	0°C to +70°C	5.0	8 PDIP (300 mils)	DS1307
DS1307N+	-40°C to +85°C	5.0	8 PDIP (300 mils)	DS1307N
DS1307Z+	0°C to +70°C	5.0	8 SO (150 mils)	DS1307
DS1307ZN+	-40°C to +85°C	5.0	8 SO (150 mils)	DS1307N
DS1307Z+T&R	0°C to +70°C	5.0	8 SO (150 mils) Tape and Reel	DS1307
DS1307ZN+T&R	-40°C to +85°C	5.0	8 SO (150 mils) Tape and Reel	DS1307N

+Denotes a lead-free/RoHS-compliant package.
**A "+" anywhere on the top mark indicates a lead-free package. An "N" anywhere on the top mark indicates an industrial temperature range device.*

ABSOLUTE MAXIMUM RATINGS

Voltage Range on Any Pin Relative to Ground .....-0.5V to +7.0V
Operating Temperature Range (Noncondensing)
Commercial.....0°C to +70°C
Industrial .....-40°C to +85°C
Storage Temperature Range ..... -55°C to +125°C
Soldering Temperature (DIP, leads).....+260°C for 10 seconds
Soldering Temperature (surface mount).....Refer to the JPC/JEDEC J-STD-020 Specification.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to the absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED DC OPERATING CONDITIONS

(T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Logic 1 Input	V_{IH}		2.2		V_{CC} + 0.3	V
Logic 0 Input	V_{IL}		-0.3		+0.8	V
V_{BAT} Battery Voltage	V_{BAT}		2.0	3	3.5	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Input Leakage (SCL)	I_{LI}		-1		1	µA
I/O Leakage (SDA, SQW/OUT)	I_{LO}		-1		1	µA
Logic 0 Output (I_{OL} = 5mA)	V_{OL}				0.4	V
Active Supply Current (f_{SCL} = 100kHz)	I_{CCA}				1.5	mA
Standby Current	I_{CCS}	(Note 3)			200	µA
V_{BAT} Leakage Current	I_{BATLKG}			5	50	nA
Power-Fail Voltage (V_{BAT} = 3.0V)	V_{PF}		1.216 x V_{BAT}	1.25 x V_{BAT}	1.284 x V_{BAT}	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 0V, V_{BAT} = 3.0V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
V_{BAT} Current (OSC ON); SQW/OUT OFF	I_{BAT1}			300	500	nA
V_{BAT} Current (OSC ON); SQW/OUT ON (32kHz)	I_{BAT2}			480	800	nA
V_{BAT} Data-Retention Current (Oscillator Off)	I_{BATDR}			10	100	nA

WARNING: Negative undershoots below -0.3V while the part is in battery-backed mode may cause loss of data.

AC ELECTRICAL CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
SCL Clock Frequency	f_{SCL}		0		100	kHz
Bus Free Time Between a STOP and START Condition	t_{BUF}		4.7			µs
Hold Time (Repeated) START Condition	$t_{HD:STA}$	(Note 4)	4.0			µs
LOW Period of SCL Clock	t_{LOW}		4.7			µs
HIGH Period of SCL Clock	t_{HIGH}		4.0			µs
Setup Time for a Repeated START Condition	$t_{SU:STA}$		4.7			µs
Data Hold Time	$t_{HD:DAT}$		0			µs
Data Setup Time	$t_{SU:DAT}$	(Notes 5, 6)	250			ns
Rise Time of Both SDA and SCL Signals	t_R				1000	ns
Fall Time of Both SDA and SCL Signals	t_F				300	ns
Setup Time for STOP Condition	$t_{SU:STO}$		4.7			µs

CAPACITANCE

(T_A = +25°C)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Pin Capacitance (SDA, SCL)	$C_{I/O}$				10	pF
Capacitance Load for Each Bus Line	C_B	(Note 7)			400	pF

Note 1: All voltages are referenced to ground.

Note 2: Limits at -40°C are guaranteed by design and are not production tested.

Note 3: I_{CCS} specified with V_{CC} = 5.0V and SDA, SCL = 5.0V.

Note 4: After this period, the first clock pulse is generated.

Note 5: A device must internally provide a hold time of at least 300ns for the SDA signal (referred to the $V_{IH(MIN)}$ of the SCL signal) to bridge the undefined region of the falling edge of SCL.

Note 6: The maximum $t_{HD:DAT}$ only has to be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal.

Note 7: C_B—total capacitance of one bus line in pF.

TIMING DIAGRAM

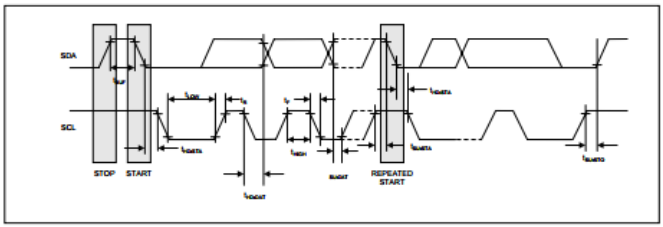
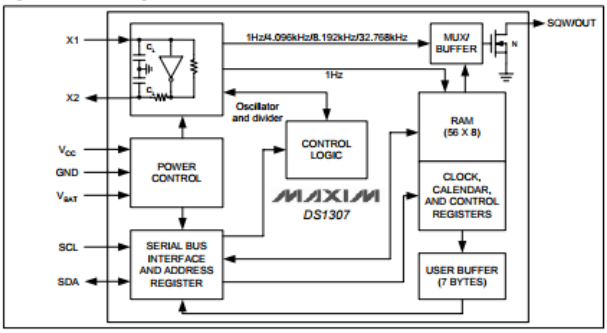


Figure 1. Block Diagram



B.4 DATASHEET RS485

RS-485 module for Arduino (MAX485)

Click photo above for details, then hover over upper right for more photos.

This module interfaces an Arduino or similar microcomputer to RS-485. RS485 is used for Serial Communications over longer distances than direct RS232 or TTL, and supports multiple units on the same bus (Multi-Drop).

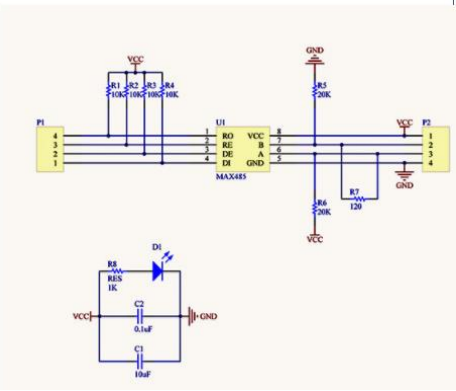
- Multiple Units can be connected to the same RS-485 wiring.
- All Chip pins are brought out for proper controls
- Working voltage: 5V
- Board size: 44 (mm) x14 (mm

THIS USB INTERFACE for use with PC's is available to connect by RS-485 (click)

How-To Information Link HERE:

See Example RS485 Network Diagram below.

SCHEMATIC OF THIS MODULE:



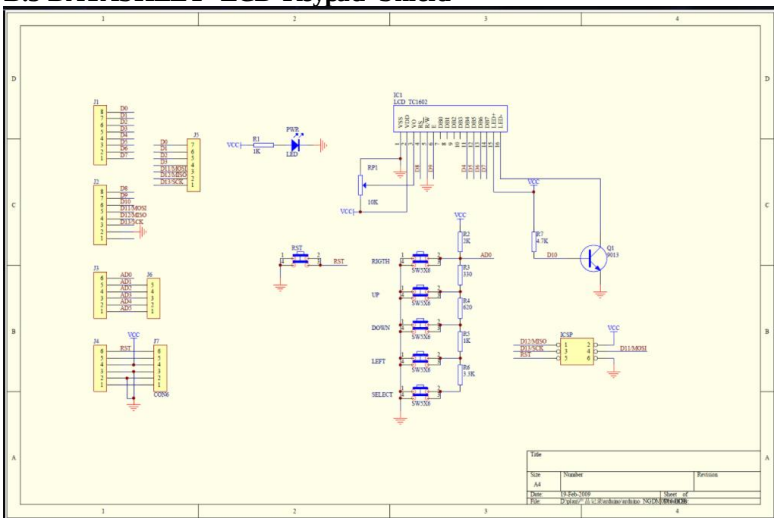
Selection Table

PART NUMBER	HALF/FULL DUPLEX	DATA RATE (Mbps)	SLEW-RATE LIMITED	LOW-POWER SHUTDOWN	RECEIVER/ DRIVER ENABLE	QUIESCENT CURRENT (µA)	NUMBER OF RECEIVERS ON BUS	PIN COUNT
MAX481	Half	2.5	No	Yes	Yes	300	32	8
MAX483	Half	0.25	Yes	Yes	Yes	120	32	8
MAX485	Half	2.5	No	No	Yes	300	32	8
MAX487	Half	0.25	Yes	Yes	Yes	120	128	8
MAX488	Full	0.25	Yes	No	No	120	32	8
MAX489	Full	0.25	Yes	No	Yes	120	32	14
MAX490	Full	2.5	No	No	No	300	32	8
MAX491	Full	2.5	No	No	Yes	300	32	14
MAX1487	Half	2.5	No	No	Yes	230	128	8

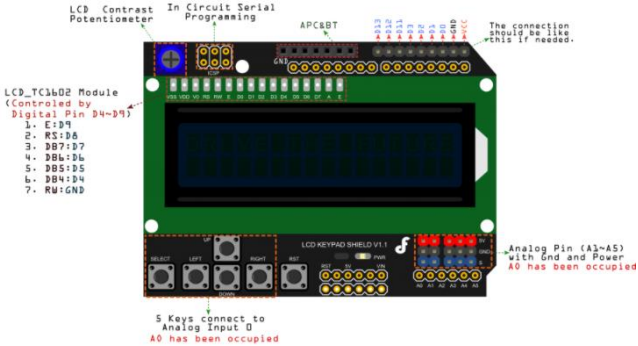
For pricing, delivery, and ordering information, please contact Maxim Direct at 1-888-629-4642, or visit Maxim Integrated's website at www.maximintegrated.com.

19-0122; Rev 10; 9/14

B.5 DATASHEET LCD Keypad Shield



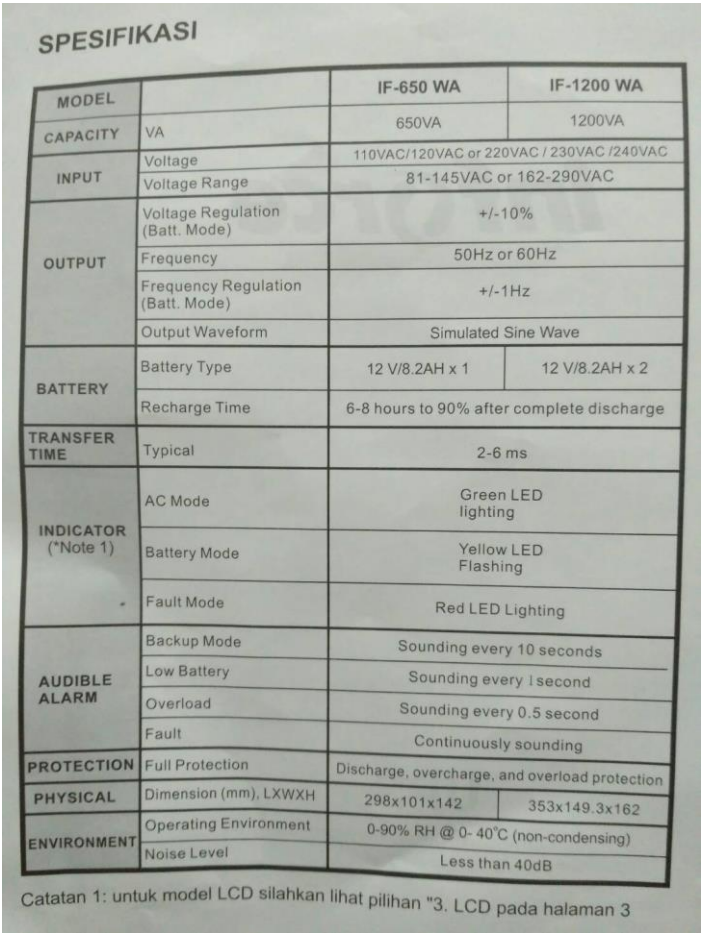
Pinout



Instruction for D4 To D10 and Analog Pin 0

Pin	Function	Instruction
Digital 4(D4)		
Digital 5(D5)	D4~D7 are used as DB4~DB7	Four high order bidirectional tristate data bus pins. Used for data transfer and receive between the MPU and the LCD.
Digital 6(D6)		
Digital 7(D7)		
Digital 8(D8)	RS	Choose Data or Signal Display
Digital 9(D9)	Enable	Starts data read/write
Digital 10(D10)	LCD Backlight Control	
Analog 0(A0)	Button select	Select, up, right, down and left

B.6. DATASHEET UPS



SPESIFIKASI

MODEL		IF-650 WA	IF-1200 WA
CAPACITY	VA	650VA	1200VA
INPUT	Voltage	110VAC/120VAC or 220VAC / 230VAC /240VAC	
	Voltage Range	81-145VAC or 162-290VAC	
OUTPUT	Voltage Regulation (Batt. Mode)	+/-10%	
	Frequency	50Hz or 60Hz	
	Frequency Regulation (Batt. Mode)	+/-1Hz	
	Output Waveform	Simulated Sine Wave	
BATTERY	Battery Type	12 V/8.2AH x 1	12 V/8.2AH x 2
	Recharge Time	6-8 hours to 90% after complete discharge	
TRANSFER TIME	Typical	2-6 ms	
INDICATOR (*Note 1)	AC Mode	Green LED lighting	
	Battery Mode	Yellow LED Flashing	
	Fault Mode	Red LED Lighting	
AUDIBLE ALARM	Backup Mode	Sounding every 10 seconds	
	Low Battery	Sounding every 1 second	
	Overload	Sounding every 0.5 second	
	Fault	Continuously sounding	
PROTECTION	Full Protection	Discharge, overcharge, and overload protection	
PHYSICAL	Dimension (mm), LXWXH	298x101x142	353x149.3x162
ENVIRONMENT	Operating Environment	0-90% RH @ 0- 40°C (non-condensing)	
	Noise Level	Less than 40dB	

Catatan 1: untuk model LCD silahkan lihat pilihan "3. LCD pada halaman 3

LAMPIRAN C
PENGUJIAN ALAT

C.1. PENUJIAN SENSOR ARUS DAN TEGANGAN

C.2. PENGUJIAN IMPEDANSI

C.3. PENGUJIAN POWER SUPPLY

C.4. PENGUJIAN RELAY UTAMA

DAFTAR RIWAYAT HIDUP

Nama	:	Dibyo Sindhu Pradana
TTL	:	Tulungagung, 10 Februari 1996
Jenis Kelamin	:	laki-laki
Agama	:	Islam
Alamat	:	Rt 03 Rw 04, Desa Ketanon, Kec Kedungwaru, Kab Tulungagug
Telp/HP	:	085607551160
E-mail	:	dibyosindhu@gmail.com

RIWAYAT PENDIDIKAN

1. 2002 – 2008 : SDN Ketanon 1 Tulungagung
2. 2008 – 2011 : SMP Negeri 1 Tulungagug
3. 2011 – 2014 : SMA Negeri 1 Boyolangu
4. 2014 – 2017 : D3 Teknik Elektro Otomasi, Program Studi Teknik Listrik – Fakultas Vokasi Institut Teknologi Sepuluh Nopember (ITS)

PENGALAMAN KERJA

1. Kerja Praktek di PT PLN (Persero) APD Jawa Timur Area Surabaya Selatan

-----Halaman ini sengaja dikosongkan-----